Министерство образования и науки Украины

В.Д. Дмитриенко, И.П. Хавина, А.Ю. Заковоротный,
М.В. Липчанский, Н.В. Мезенцев

Методы и алгоритмы систем искусственного интеллекта

Рекомендовано Министерством образования и науки Украины
как учебное пособие

Харьков НТУ «ХПИ» 2014

УДК 004.89 (072)
ББК 32.813 Я73
М 54

Рецензенты:
Ю.В. Батыгин, д-р техн. наук, проф., Харьковский национальный автомобильно-дорожный университет;
Г.И. Загарий, д-р техн. наук, проф., Украинская государственная академия железнодорожного транспорта

Авторский коллектив: В.Д. Дмитриенко, И.П. Хавина, А.Ю. Заковоротный, М.В. Липчанский, Н.В. Мезенцев

Рекомендовано Министерством образования и науки Украины, письмо № от 2014 г.
как учебное пособие для студентов высших учебных заведений

Навчальний посібник включає основні відомості щодо методів та алгоритмів створення інтелектуальних систем. Наведені приклади демонструють різні підходи до побудови систем штучного інтелекту.

Призначено для фахівців у галузі інформатики і штучного інтелекту, студентів та аспірантів вузів.

Методы и алгоритмы систем искусственного интеллекта: учеб. пособие /
М 54 В.Д. Дмитриенко, И.П. Хавина, А.Ю. Заковоротный и др. – Х.: НТУ «ХПИ», 2014. – 272 с. На русск. яз.

ISBN

Учебное пособие включает основные сведения по методам и алгоритмам создания интеллектуальных систем. Приведенные примеры учебных задач дают представление о различных подходах к построению систем искусственного интеллекта.

Предназначено для специалистов в области информатики и искусственного интеллекта, студентов и аспирантов вузов.

Ил. 41. Табл. 18. Библиогр.:32 назв.

УДК 004.89 (072)
ББК 32.813 Я73

ISBN

СОДЕРЖАНИЕ

Список сокращений

БД – база данных

БЗ – база знаний

ИА – интеллектуальный агент

ИИ – искусственный интеллект

ИПС – информационно-поисковая система

ИС – интеллектуальная система

ИНС – искусственная нейронная сеть

НС – нейронная сеть

ПК – персональный компьютер

РНК – рибонуклеиновая кислота

РП – рабочая память

СУБД – система управления базами данных

СППР – система поддержки принятия решений

ТДШ – теория свидетельств Демпстера-Шефера

ЭС – экспертная система

ИА – интеллектуальный агент

МАС – мультиагентная система

ФКВ – функция коллективного выбора

AIS – алгоритм поиска ассоциативных правил

SQL – язык обработки запросов

OLAP – On-Line Analytical Processing

Все новые идеи проходят три стадии. На первой идея кажется нелепой, на второй встречает яростное сопротивление, а на третьей воспринимается как нечто очевидное.

Артур Шопенгауэр

ПРЕДИСЛОВИЕ

Единого ответа на вопрос, чем занимается искусственный интеллект (ИИ), не существует. Почти каждый автор, пишущий книгу об ИИ, отталкивается в ней от какого-либо определения, рассматривая в его свете достижений этой науки. Данное учебное пособие посвящено описанию методов и алгоритмов искусственного интеллекта, которые показали хорошие результаты в различных областях науки и технике.

Раздел 1 вводит в понятия систем ИИ и в основные направления в разработке систем. Дана общая структура систем искусственного интеллекта.

Распознаванию образов посвящен раздел 2, где показаны методы решения широкого круга задач из этой области.

Системы ИИ отличаются от других систем способностью к обучению. В разделе 3 показаны некоторые способы построения обучающихся систем.

Раздел 4 посвящен построению экспертных систем, которые основаны на знаниях, и содержит способы представления знаний и логику предикатов.

Построение систем ИИ в условиях неопределенности дано в разделе 5, а раздел 6 знакомит с простыми искусственными нейронными сетями.

Эволюционные методы ИИ даны в разделе 7, где кратко описаны метод эволюционного моделирования, метод группового учета аргументов и генетические алгоритмы, а раздел 8 содержит классификацию стадий Data mining, описание некоторых технологий и пример построения деревьев решений на основе ассоциативных правил.

Агентно-ориентированный подход для моделирования распределенных динамических систем показан в разделе 9.

Техника дойдет до такого совершенства, что человек сможет обойтись без самого себя.

Станислав Ежи Лец, польский поэт

1. ВВЕДЕНИЕ В ПОНЯТИЯ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

“Искусственный интеллект (ИИ) – это наука о концепциях, позволяющих компьютеру осуществлять действия, которые у людей считаются разумными. Интеллект человека представляет собой способность размышлять, усваивать и использовать знания, обмениваться идеями” – такое определение дал П. Уинстон в книге “Искусственный интеллект” изданной в 1984. Все эти способности являются частью того, что называется интеллектом. В настоящее время известно около 70 определений ИИ. Наиболее подходящим для этого пособия является следующее.

Искусственный интеллект (с лат. *intellectus* – познание) раздел информатики, изучающий с помощью компьютера методы, способы моделирования и воспроизведения деятельности человека, связанной с решением различных задач [1].

1.1. Основные направления в разработке интеллектуальных систем

Общую структуру исследований в искусственном интеллекте можно представить различными схемами, рассмотрим одну из них, изображенную на рис. 1.1, где выделяются два основных направления в исследованиях ИИ: бионическое и программно-прагматическое.

В бионическом направлении выделяют *три различных подхода* [2]. *Пер-*

вый из них – *нейробионический*. В его основе лежат системы нейроподобных элементов, из которых создаются системы, способные воспроизводить некоторые интеллектуальные функции. К числу задач, которые, по-видимому, могут быть решены в рамках этого подхода, относится многоканальное (параллельное) распознавание сложных зрительных образов, обучение условным рефлексам и др.

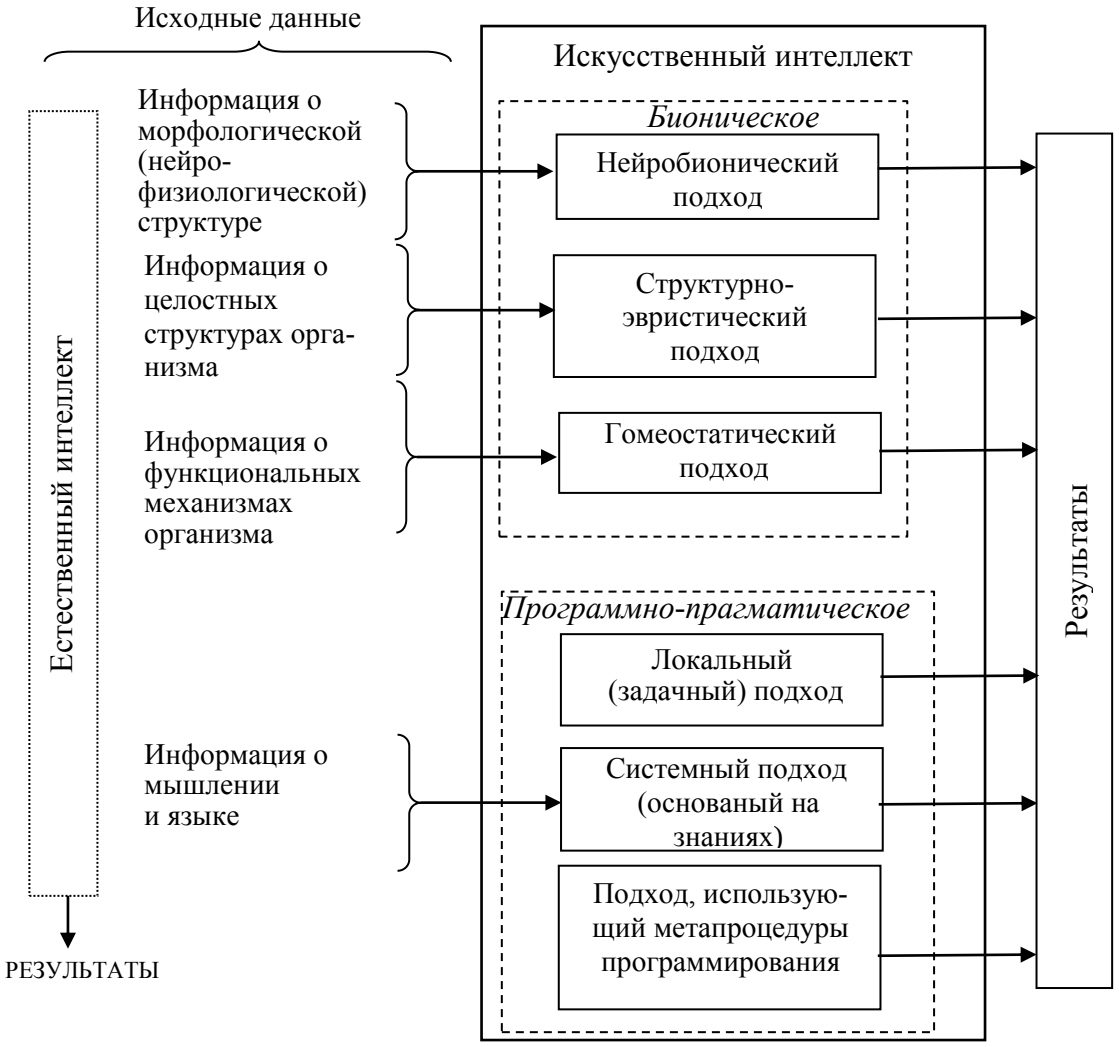


Рис. 1.1. Структура исследований в искусственном интеллекте

Второй подход – структурно-эвристический. В его основе лежат знания о наблюдаемом поведении объекта, рассматриваемого как “черный ящик”, и соображения о тех структурах (и их свойствах) мозга, которые могли бы обес-

печить реализацию наблюдаемых форм поведения.

Третий подход– гомеостатический. В этом случае мозг рассматривается как гомеостатическая система, представляющая собой совокупность противоборствующих (и сотрудничающих) подсистем, в результате функционирования которых обеспечивается нужное равновесие (устойчивость) всей системы в условиях постоянно изменяющихся воздействий среды.

Гомеостатические модели подтверждают перспективность этого подхода. Однако в настоящее время еще не существует гомеостатических модулей, которые могли бы рассматриваться как универсальные элементы для создания интеллектуальных систем.

В бионическом направлении наибольшее развитие получили нейронные сети, с помощью которых решено огромное количество практических задач и осуществляются попытки промоделировать строение и функционирование человеческого мозга [2].

В программно-прагматическом направлении ИИ можно выделить три подхода.

Первый подход – локальный – основан на точке зрения, что для каждой задачи, присущей творческой деятельности человека, можно найти способ ее решения на компьютере, который, будучи реализован в виде программы, даст результат, либо подобный результату, полученному человеком, либо даже лучший. Разработано много программ такого рода. Типичным примером являются шахматные программы, которые играют в шахматы лучше большинства людей, но в основе их лежат идеи, далекие от тех, которыми пользуются люди [2].

Второй подход – системный, или основанный на знаниях, связан с представлением о том, что решение отдельных творческих задач не исчерпывает всей проблематики искусственного интеллекта. Естественный интеллект человека способен не только решать творческие задачи, но при необходимости обучаться тому или иному виду творческой деятельности. Поэтому и программы искусственного интеллекта должны быть ориентированы не только или не

столько на решение конкретных интеллектуальных задач, сколько на создание средств, позволяющих автоматически строить программы решения интеллектуальных задач, когда в таких программах возникает необходимость. Этот подход в настоящее время является центральным в программно-прагматическом направлении [2].

Третий подход рассматривает проблемы создания интеллектуальных систем как часть общей теории программирования (как некоторый новый виток в этой теории). При этом подходе для составления интеллектуальных программ используются обычные программные средства, позволяющие писать нужные программы по описаниям задач на профессиональном естественном языке. Все метасредства, возникающие при этом на базе частичного анализа естественного интеллекта, рассматриваются здесь лишь с точки зрения создания интеллектуального программного обеспечения, то есть комплекса средств, автоматизирующих деятельность самого программиста [2].

С точки зрения конечного результата в программно-прагматическом направлении выделяются четыре больших раздела:

1) интеллектуальные программы (для решения интеллектуальных задач);
2) работа со знаниями (теория и программы);
3) интеллектуальное программирование (теория и сервисные интеллектуальные программы);
4) интеллектуальные программные системы.

Интеллектуальные программы разбиваются на несколько групп и подгрупп, определяемых типами задач, решаемых этими программами:

- ✓ игровые программы: человеческие и компьютерные игры;
- ✓ естественно-языковые программы: естественно-языковой интерфейс, машинный перевод, автоматическое реферирование, синтез текстов;
- ✓ музыкальные программы: сочинение, анализ и имитация музыкальных произведений;
- ✓ распознающие и узнающие программы;

✓ программы создания произведений графики и живописи;

✓ прочие программы: модели поведения; программы доказательства теорем; эвристические программы.

Работа со знаниями лежит в основе современного периода развития искусственного интеллекта [2]. Всякая предметная (проблемная) область деятельности может быть описана в виде совокупности сведений о структуре этой области, основных ее характеристиках, процессах, протекающих в ней, а также о способах решения возникающих в ней задач. Все эти сведения образуют знания о предметной области. Для решения задач в данной предметной области необходимо собрать знания о ней и создать концептуальную модель этой области. Источниками знаний могут быть документы, фотографии и многое другое.

Из этих источников надо извлечь содержащиеся в них знания. Этот процесс оказывается достаточно трудным, т.к. надо заранее оценить важность и нужность тех или иных знаний для работы интеллектуальной системы.

В качестве основных средств представления знаний в интеллектуальных системах в настоящее время используются системы продукций, системы фреймов, семантические сети и системы ограничений. Каждое из выделенных средств представления ориентировано на описание разных типов знаний и обладает разными возможностями и свойствами.

В области извлечения знаний выделяются два основных раздела: формализация качественных знаний и интеграция знаний. Первый связан с созданием методов, позволяющих переходить от знаний, выраженных в текстовой форме, к их аналогам, пригодным для ввода в память интеллектуальной системы.

В связи с этой проблемой развивались не только традиционные методы обработки экспериментальных данных, но и новое направление, получившее название нечеткой математики. Нечеткая математика и ее методы оказали существенное влияние на многие области искусственного интеллекта, и, в частности, на весь комплекс проблем, связанный с представлением и переработкой качественной информации.

Полученные от экспертов знания нужно оценить с точки зрения их соответствия ранее накопленным знаниям и формализовать для ввода в память системы. Кроме того, знания, полученные от различных экспертов, надо согласовать между собой, устранить в них противоречия путем опроса.

Следующая проблема – это представление знаний в памяти системы. В настоящее время в интеллектуальных системах используются *три основные модели знаний*.

Первая модель, возможно, наиболее близка к тому, как представляются знания в текстах на естественном языке. В ее основе лежит идея о том, что вся необходимая информация может быть описана как совокупность троек вида (arb), где a и b – два объекта или понятия, а r – бинарное отношение между ними. Такая модель графически может представляться в виде сети, в которой вершинам соответствуют объекты или понятия, а дугам – отношения между ними. Дуги помечены именами соответствующих отношений. Эта модель носит название *семантической сети*.

Под *фреймами* понимают описания вида “Имя фрейма (Множество слотов)”. Каждый слот есть пара вида (Имя слота. Значение слота). Допускается, чтобы слот сам был фреймом. Тогда в качестве значений слота выступает множество слотов. Таким образом, фрейм представляет собой гибкую конструкцию, позволяющую отображать в памяти интеллектуальной системы разнообразные знания.

Третья распространенная модель знаний опирается на классическую *логическую* модель вывода. Это либо логические исчисления типа исчисления предикатов и его расширений, либо системы продукций, задающих элементарные шаги преобразований или умозаключений.

В основе подхода, использующего метапроцедуры программирования или *интеллектуального программирования* лежит создание инструментария, ориентированного на поддержку разработки интеллектуальных систем:

1) языки для искусственного интеллекта: языки логического программи-

рования; объектно-ориентированные языки; языки для представления знаний;

2) автоматический синтез программ: дедуктивные и индуктивные методы;

3) инструментальные системы: “пустые” системы; оболочки;

4) системы когнитивной графики;

5) агентно-ореентированный подход.

Лишь небольшая часть языков программирования ориентирована на задачи искусственного интеллекта. Так, например, язык Лисп отражает ту точку зрения, что основой большинства интеллектуальных задач являются хорошо организованные перебор и поиск, а логический вывод породил язык Пролог.

Представление о том, что процедуры логического вывода в задачах искусственного интеллекта должны быть дополнены новой конструкцией, в основе которой лежит объект с его свойствами и признаками, привело к появлению объектно-ориентированных языков. При этом решение задач представляется как манипулирование с понятиями, обобщающими объекты, которые связаны с проблемной областью [2].

Типично программистский характер имеют и работы по автоматизации программирования. Синтез программ может быть осуществлен из типовых блоков-модулей по описанию исходной задачи в рамках некоторой *дедуктивной* системы. В этом случае процедура синтеза представляет собой нечто вроде логического вывода, в ходе которого программа извлекается по ходу вывода.

Другой вид синтеза – *индуктивный* – представляет собой процесс генерирования программы в ходе обучения на множестве примеров. Основной трудностью здесь является выбор способа формального описания функциональных особенностей и свойств синтезируемых программ [2].

Специфическим разделом интеллектуального программирования являются системы когнитивной графики, которые пытаются реализовать основную идею современного представления о мышлении как о синтезе визуальных и символьных представлении о внешнем мире [2].

Исторически *символьный подход* в системах ИИ был реализован первым.

Символьный подход позволяет оперировать слабоформализованными представлениями и их смыслами.

Затем появился *логический подход* к созданию систем ИИ, который направлен на создание экспертных систем с логическими моделями баз знаний с использованием языка предикатов.

С 1990-х годов развивается *агентно-ориентированный подход*, основанный на использовании интеллектуальных агентов. Под *интеллектуальным агентом* (ИА) понимают программно или аппаратно реализованную систему, которая обладает свойствами: автономности (функционирует без вмешательства человека); общественным поведением (сосуществует с другими агентами, обмениваясь с ними сообщениями); реактивностью (воспринимает среду и реагирует на ее изменения); про-активностью (генерирует цели и действует для их достижения).

Агент обладает:

✓ знаниями – это знания агента о себе, среде и других агента, которые не изменяются в процессе функционирования;

✓ убеждениями – знания агента о среде, о других агентах. Это те знания, которые могут изменяться во времени или становится неверными;

✓ желания – это состояния, ситуации, достижение которых для агента желательно, однако они могут быть противоречивыми и потому агент не ожидает, что все они будут достигнуты;

✓ намерениями – это то, что агент или обязан сделать в силу своих обязательств по отношению к другим агентам, или то, что вытекает из его желаний;

✓ целями – конкретное множество конечных и промежуточных состояний, достижение которые агент принял в качестве текущей стратегии поведения;

✓ обязательствами по отношению к другим агентам – задачи, которые агент берет на себя по просьбе (поручению) других агентов в рамках кооперативных целей или целей отдельных агентов в рамках сотрудничества.

Идея использования программных агентов оказалась настолько удачной,

что получила свое развитие. Качественный переход произошел в следующем направлении: агент может не только общаться с пользователем, интересы которого он представляет, но и с другими агентами, которые могут представлять интересы других пользователей, то есть система стала многоагентной – *мультиагентной*.

1.2. Структура интеллектуальной системы

Общая структура интеллектуальной программной системы изображена на рис. 1.2.

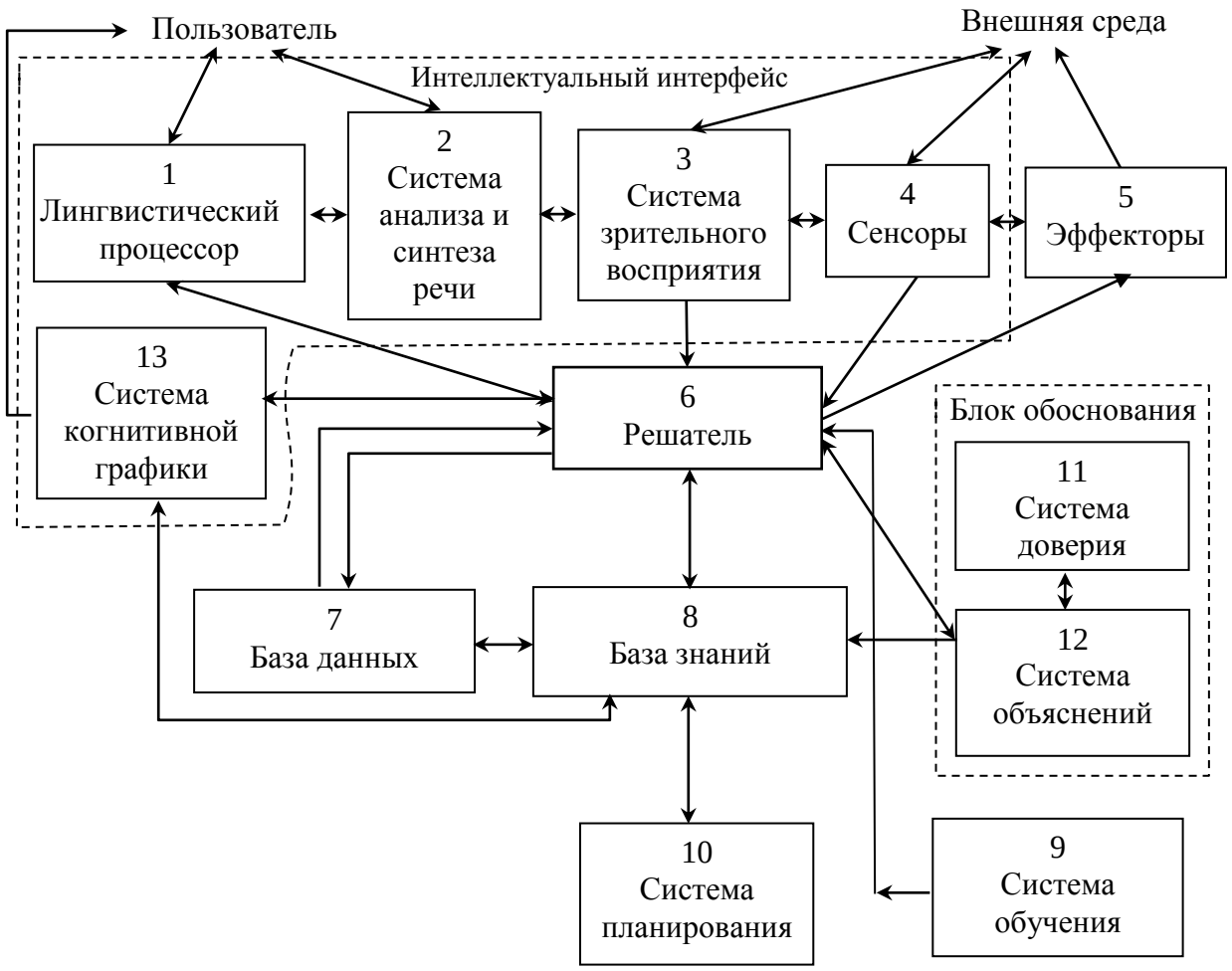


Рис. 1.2. Структура интеллектуальной системы

Система содержит 13 функциональных блоков, часть которых может быть объединена в функциональные группы. Одной такой группой является интеллектуальный интерфейс, обеспечивающий эффективную связь всей системы с пользователем и внешней средой [2]. В состав интеллектуального интерфейса могут входить блоки 1 – 4 и 13.

Лингвистический процессор обеспечивает связь пользователя с системой на естественном ограниченном языке: ввод и понимание системой текстов на нем и вывод текстов, вырабатываемых системой.

Для голосового общения пользователя с системой используется система анализа и синтеза речи. Информация из внешней среды воспринимается системой с помощью сенсоров. При этом зрительная информация перед поступлением в систему обрабатывается в системе зрительного восприятия. Если система имеет возможность воздействовать на внешнюю среду, то в состав интеллектуального интерфейса должен быть включен блок эффекторов (манипуляторов).

Система когнитивной графики позволяет пользователю воспринимать результаты работы системы в графической форме и общаться на языке графики.

Центральным блоком ИС является решатель – вычислительная система, состоящая из одного или нескольких компьютеров (процессоров), связанная с базами данных и знаний, а также с остальными блоками системы.

Целенаправленная работа системы обеспечивается системой планирования, хранящей априорно введенные цели, а также запоминающей новые цели, полученные с помощью системы обучения, которая участвует в формировании новых знаний, возникающих в ходе взаимодействия ИС с внешней средой.

Группа блоков обоснования, включающих систему объяснения и систему доверия, служит для обоснования полученных системой решений с привлечением информации, содержащейся в базе данных. Все перечисленные блоки, за исключением блоков 4 и 5, могут быть реализованы как на специальных аппаратных средствах, так и в решателе с использованием его логико-вычислительных возможностей.

Кроме того, в зависимости от степени развития и функциональных возможностей конкретных ИС в их структуру часть перечисленных блоков может

не входить [2].

Для установления соответствия между конкретными функциональными структурами основных типов интеллектуальных систем и типовой схемой рис. 1.2 рассмотрим табл. 1.1.

Таблица 1.1 – Состав интеллектуальной системы

Вид интеллектуальной системы	Номер блока												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Экспертная	+	0	–	–	–	+	+	+	0	0	+	+	0
Информационная	+	0	–	–	–	+	+	+	–	–	–	–	–
Расчетно-логическая (гибридная)	+	–	–	–	–	+	+	+	0	+	–	–	+
Проектирование научных исследований	+	–	0	–	–	+	+	+	–	+	0	0	+
Обучающая	+	0	–	–	–	+	+	+	+	+	+	+	–
Интеллектуальные роботы	+	+	+	+	+	+	+	+	0	+	0	0	0

В ней для каждого вида интеллектуальной системы показано, какие блоки в этот вид обязательно входят (+) и какие не входят (–). Нулями отмечены блоки, которые могут входить или не входить в систему в зависимости от характера решаемых задач и технического совершенства системы [2].

Дальнейшие перспективы развития искусственного интеллекта будут связаны с использованием результатов достижений в разработке квантовых компьютеров с применением Grid-технологии, а также биотехнологии WETLAB, в которой используются нити РНК.

Однако понятие *Интеллекта* как способностей *человеческого Разума* и *Души* генерировать новые решения остается неопределенным. Если медико-биологические и психофизиологические процессы жизнедеятельности человека изучаются на все более детальном уровне, то закономерности и свойства интеллекта и его способности к принятию решений в нечетких и трудно формализуемых ситуациях все еще остаются туманными, не доступными для физиологов, психологов и системных аналитиков.

Контрольные вопросы

1. Каковы основные направления в разработке ИИ?
2. Перечислите задачи, решаемые с помощью методов ИИ.
3. Какова структура интеллектуальной программной системы?
4. Какие есть подходы к созданию систем ИИ?
5. Перечислите свойства агентно-ориентированной системы.
6. Что такое мультиагентная система?
7. Перечислите свойства интеллектуального агента.
8. Какова структура экспертной системы?
9. Чем ИС отличаются от всех других систем?
10. Какова структура обучающейся системы?
11. Каковы перспективы систем искусственного интеллекта?
12. Каковы функции лингвистического процессора в системе ИИ?
13. Каковы функции системы когнитивной графики?
14. Для чего нужны системы объяснения и доверия?
15. Что входит в центральный блок интеллектуальной системы?
16. Для чего нужна система анализа и синтеза речи?

Порядок – это хаос, к которому привыкли.

Роберт Лембке, немецкий журналист

2. ВВЕДЕНИЕ В РАСПОЗНАВАНИЕ ОБРАЗОВ

Распознавание образов является важным разделом искусственного интеллекта. В настоящее время это – сложившиеся научное и практическое направления, связанные с решением широкого круга задач, относящихся к проблемам распознавания [3].

2.1. Образы и распознавание образов

Образы, с которыми мы встречаемся, можно разделить на две категории: абстрактные и конкретные. Примером абстрактных образов могут быть идеи и аргументы. Распознавание таких образов относится к концептуальному распознаванию, которое в данном курсе не рассматривается [3].

Примером конкретного распознавания является распознавание букв, символов, рисунков, биологических изображений, трехмерных физических объектов, речевых сигналов, электрокардиограмм, сейсмических волн. Некоторые из этих образов – пространственные, другие – временные [3].

Воспринимаемые объекты (например буквы, символы, сигналы) разбиваются человеком на группы или классы похожих, обладающих общими свойствами, но в тоже время отличающиеся некоторыми второстепенными свойствами, неважными с точки зрения решаемой задачи. Множество таких объектов называют образами. Образ – своего рода представитель некоторого класса

объектов. С этой точки зрения образ является абстрактным понятием, соответствующим не одному а множеству объектов. Это множество объектов можно рассматривать и как различное проявление образа.

Последние два десятилетия основной интерес был направлен на два основных типа проблем распознавания.

1) Механизм распознавания, осуществляемый живыми организмами. Психологи, физиологи, биологи и нейрофизиологи приложили много усилий для изучения того, как живые организмы воспринимают объекты. Многое из результатов этих исследований отражено в литературе по бионике и другим близким областям.

2) Развитие теории и практики компьютерного решения данной задачи.

Это направление, в котором активно работают как инженеры, так и прикладные математики. Здесь нет единой теории, которая позволила бы решить все виды задач распознавания образов. Большинство методов имеют проблемную ориентацию и примеры решения конкретных задач, связанных с распознаванием изображений и речевых сигналов [3].

2.2. Задача распознавания образов

В распознавании образов разделяют общую задачу на три фазы: получение данных; предварительная обработка данных; принятие решения о классификации [3]. В фазе получения данных аналоговые данные из физического мира собираются с помощью соответствующих датчиков и преобразователей и превращаются в цифровой формат для компьютерной обработки. На этом этапе физические переменные превращаются в ряд измеримых данных. Измеренные данные затем используются на второй фазе (предобработка данных) и образуют ряд классификационных признаков на выходе. Третья фаза – это принятие решения с помощью классификатора, основой которого является решающие функции [3].

2.3. Представление изображений при машинном распознавании

Существует большое число различных форм представления изображений в распознающих устройствах или программах. Одной из наиболее простых и понятных является форма, использующая представление изображений в виде точек в некотором *n*-мерном пространстве. Каждая ось такого пространства естественным образом соотносится с одним из *n* входов или с одним из *n* рецепторов распознающей системы. Каждый из рецепторов может находиться в одном из *m* состояний, если они дискретны, или иметь бесконечно большое число состояний, если рецепторы непрерывны. В зависимости от вида используемых рецепторов может порождаться непрерывное, дискретное или непрерывно-дискретное *n*-мерное пространство.

Как правило, в пространстве изображений вводится метрика – функция, которая каждой упорядоченной паре точек *x* и *y* пространства ставит в соответствие действительное число $d(x, y)$. При этом функция $d(x, y)$ обладает следующими свойствами:

1) $d(x, y) \geq 0$, $d(x, y) = 0$ тогда и только тогда, когда $x = y$;

2) $d(x, y) = d(y, x)$;

3) $d(x, y) \leq d(x, z) + d(z, y)$.

Введение метрики $d(x, y)$ в пространстве изображений позволяет говорить о близости или удаленности точек в этом пространстве или о мере сходства или различия анализируемых изображений. Понятие меры сходства изображений широко используется в теории распознавания образов. Однако формализация этого понятия при решении конкретных задач распознавания, как правило, не является тривиальной задачей. Более того, эта задача является одной из основных задач теории распознавания образов. Рассмотрим общие требования к мере сходства изображений.

Пусть задано некоторое конечное множество $S = \{S_1, S_2, \ldots, S_n\}$ входных изображений, каждое из которых является точкой в n-мерном пространстве изображений. Меру сходства изображений можно ввести как функцию двух аргументов $L(S_k, S_i)$, где $S_k, S_i \in S$. Общие требования к этой функции можно свести к следующему:

1) функция $L(S_k, S_i)$ должна обладать свойством симметрии, т.е.

$$L(S_k, S_i) = L(S_i, S_k);$$

2) область значений функции $L(S_k, S_i)$ – множество неотрицательных чисел, т.е.

$$L(S_k, S_i) \geq 0, \ k, i = 1, 2, \ldots, n;$$

3) мера сходства изображения с самим собой должна принимать экстремальное значение по сравнению с любым другим изображением, т.е. в зависимости от способа введения меры сходства должно выполняться одно из двух соотношений:

$$L(S_k, S_k) = \max_i L(S_k, S_i),$$

$$L(S_k, S_k) = \min_i L(S_k, S_i);$$

4) в случае непрерывного n-мерного пространства и компактных образов функция $L(S_k, S_i)$ должна быть монотонной функцией удаления точек S_k и S_i друг от друга в этом пространстве.

Анализ свойств метрики и меры сходства изображений показывает, что требования к функции $L(S_k, S_i)$ нетрудно выполнить в метрических пространствах. В частности, если в метрическом пространстве введено расстояние, то оно может быть использовано в виде меры сходства изображений.

2.4. Распознавание по расстояниям в *n*-мерном пространстве

Эталонные изображения $X_1, X_2, \ldots, X_m$ некоторого числа m различных классов изображений или образов в n-мерном пространстве задаются в виде точек $(x_{11}, x_{12}, \ldots, x_{1n}), (x_{21}, x_{22}, \ldots, x_{2n}), \ldots, (x_{m1}, x_{m2}, \ldots, x_{mn})$. Любое входное изображение S_i также представляется в виде точки $(x_{Si1}, x_{Si2}, \ldots, x_{Sin})$ в этом пространстве. Принадлежность входного изображения S_k к одному из m классов определяется с помощью расстояний между точкой S_i и всеми точками $X_1, X_2, \ldots, X_m$, соответствующими эталонным изображениям образов. Расстояние и является мерой сходства входного изображения с эталонами классов или образов. Входное изображение относится к тому образу, расстояние до эталонного изображения которого минимально, т.е. решающим правилом является следующее соотношение

$$S_i \in X_j, \text{ если } L(S_i, X_j) = \min_j (L(S_i, X_j)). \tag{2.1}$$

В теории распознавания образов часто используются расстояния по Евклиду (2.2) и по Минковскому (2.3):

$$L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} (s_{ik} - x_{jk})^2}\,; \tag{2.2}$$

$$L(S_i, X_j) = \sqrt[\lambda]{\sum_{k=1}^{n} (s_{ik} - x_{jk})^{\lambda}}\,. \tag{2.3}$$

где λ – целое положительное число, большее двух.

Операции возведения в степень и извлечения корня не всегда удобно использовать при определении расстояний, поскольку они являются нелинейными операциями. Поэтому для определения расстояний в пространстве изображений часто используется и сумма модулей разностей между соответствующими компонентами n-мерных векторов:

$$L(S_i, X_j) = \sum_{k=1}^{n} \left| s_{ik} - x_{jk} \right|. \quad (2.4)$$

В выражения (2.2) – (2.4) разности всех компонентов векторов входят с одинаковыми единичными весами. В тех случаях, когда компоненты векторов, соответствующих распознаваемым изображениям, отличаются на порядки, например, одни компоненты векторов измеряются метрами, а другие – сантиметрами или миллиметрами, то при использовании расстояний (2.2) – (2.4) компоненты, имеющие небольшие численные значения, могут практически не влиять на величину расстояний.

В то же время с точки зрения решения реальных задач распознавания именно эти компоненты могут играть определяющую роль. Поэтому для более адекватного учета подобных компонент в выражения (2.2) – (2.4) могут вводиться весовые коэффициенты, учитывающие практическую ценность различных компонент вектора. В этом случае выражения (2.2) – (2.4) преобразуются к виду:

$$L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^2}, \quad (2.5)$$

$$L(S_i, X_j) = \sqrt[\lambda]{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^{\lambda}}, \quad (2.6)$$

$$L(S_i, X_j) = \sum_{k=1}^{n} \eta_k \left| s_{ik} - x_{jk} \right|. \quad (2.7)$$

Предварительное задание весовых коэффициентов в формулах, определяющих расстояния, требует наличия определенной априорной информации и не всегда может быть сделано оптимальных образом. Поэтому особый интерес представляют расстояния, в которых заложена идея выравнивания весов слагаемых от различных компонент, если они существенно отличаются по своим аб-

солютным значениям. Примером такого расстояния является расстояние по Камберру:

$$L(S_i, X_j) = \sum_{k=1}^{n} \frac{\left| s_{ik} - x_{jk} \right|}{\left| s_{ik} + x_{jk} \right|}. \tag{2.8}$$

При решении задач распознавания сигналов часто возникают ситуации, когда сигналы, отличающиеся только амплитудой или смещением по оси ординат, или небольшими нелинейными искажениями, необходимо относить к одному классу. В этом случае может использоваться расстояние по Кендалу:

$$L(S_i, X_j) = 1 - \frac{2}{n(n-1)} \sum_{q=1}^{n-1} \sum_{\substack{k=2, \\ k>q}}^{n} \Delta_{qk}^{i} \Delta_{qk}^{j}, \tag{2.9}$$

где

$$\Delta_{qk}^{i} = \begin{cases} 1, \text{ если } x_{iq} > x_{ik}, \\ -1, \text{ если } x_{iq} < x_{ik}, \\ 0, \text{ если } x_{iq} = x_{ik}. \end{cases} \quad q < k,$$

Если компоненты обоих векторов S_i и X_j упорядочены однотипно, то $\Delta_{qk}^{i} = \Delta_{qk}^{j}$, $q = \overline{1, (n-1)}$, $k = \overline{2, n}$, $q < k$, и результат суммирования в выражении (2.9) равен половине числа размещений из n по два: $A_n^2 / 2 = n(n-1)/2$. Отсюда следует, что выражение (2.9) при однотипном упорядочении векторов S_i и X_j принимает минимальное значение, равное нулю:

$$L(S_i, X_j) = 1 - \frac{2}{n(n-1)} \frac{n(n-1)}{2} = 1 - 1 = 0.$$

Если $\Delta_{qk}^{i} \neq \Delta_{qk}^{j}$ ни при одном значении индексов q и k, то любое произведение $\Delta_{qk}^{i} \Delta_{qk}^{j} = -1$, $q = \overline{1, (n-1)}$, $k = \overline{2, n}$, $q < k$, и, следовательно, в этом

случае $L(S_i, X_j) = 1 - (-1) = 2$. Таким образом, расстояние по Кендалу может принимать значения из интервала [0, 2]. Расстояние по Кендалу – это и расстояние для оценки близости в некотором смысле двух функций, заданных в n точках. Для оценки близости двух функций $F(x)$ и $G(x)$, заданных векторами своих значений в nточках часто применяется и расстояние по Чебышеву:

$$L(F(x_i), G(x_i)) = \max_i |F(x_i) - G(x_i)|. \qquad (2.10)$$

2.5. Распознавание по углу между векторами

Мера близости между двумя векторами в n-мерном векторном пространстве может быть задана в виде угла. Если задано входное изображение $S_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ и векторы эталонных изображений $X_1 = (x_{11}, x_{12}, \ldots, x_{1n})$, $X_2 = (x_{21}, x_{22}, \ldots, x_{2n})$, …, $X_m = (x_{m1}, x_{m2}, \ldots, x_{mn})$, то мера сходства между входным и эталонными изображениями определяется выражением

$$L(S_i, X_j) = \arccos\left(\frac{s_{i1}x_{j1} + s_{i2}x_{j2} + \ldots + s_{in}x_{jn}}{\sqrt{s_{i1}^2 + s_{i2}^2 + \ldots + s_{in}^2} \cdot \sqrt{x_{j1}^2 + x_{j2}^2 + \ldots + x_{jn}^2}}\right) =$$

$$= \arccos\left(\frac{\sum_{k=1}^{n} s_{ik}x_{jk}}{|S_i| \cdot |X_j|}\right), \qquad (2.11)$$

где $|S_i|$, $|X_j|$ – соответственно длины векторов S_i и X_j.

Принадлежность входного изображения S_i к одному из m образов определяется с помощью решающего правила

$$S_i \in X_j, \text{ если } L(S_i, X_j) = \min_j L(S_i, X_j). \qquad (2.12)$$

При этом в решающем правиле и далее по тексту для обозначения *j*-го образа и эталонного изображения *j*-го образа применяется одно и тоже обозначение X_j $(j = \overline{1, m})$.

2.6. Распознавание изображений по скалярному произведению

Мера близости изображений по углу между векторами (2.11) основана на скалярном произведении векторов:

$$< S_i,\ X_j > = |S_i||X_j|\cos\alpha = \sum_{k=1}^{n} s_{ik} x_{jk}. \qquad (2.13)$$

Некоторые системы распознавания используют непосредственно скалярное произведение в качестве меры сходства изображений в *n*-мерном векторном пространстве

$$L(S_i, X_j) = \sum_{k=1}^{n} s_{ik} x_{jk}. \qquad (2.14)$$

В этом случае принадлежность входного изображения S_i к какому-либо образу определяется с помощью решающего правила

$$S_i \in X_j, \text{ если } L(S_i, X_j) = \max_j L(S_i, X_j). \qquad (2.15)$$

2.7. Распознавание изображений по принадлежности к заданной области пространства

При этом способе распознавания все пространство изображений *V* разбивается на непересекающиеся области $V_1, V_2, \ldots, V_m, V_{m+1}$, где $V_1, V_2, \ldots, V_m$ – области, содержащие изображения только одного соответствующего образа X_1,

X_2, …, X_m; V_{m+1} – область, не содержащая изображений, относящихся к указанным образам. В этом случае принадлежность входного изображения $S_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ к некоторому j-му образу $(j = \overline{1, m})$ определяется решающим правилом

$$S_i \in X_j, \text{ если } (s_{i1}, s_{i2}, ..., s_{in}) \in V_j. \tag{2.16}$$

Если области V_j $(j = \overline{1, m})$ заданы в евклидовом пространстве в виде шаров с центрами в точках $(x^*_{j1}, x^*_{j2}, \ldots, x^*_{jn})$ и радиусами R_j, то решающее правило (2.16) принимает вид

$$S_i \in X_j, \text{ если } L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} (x^*_{jk} - s_{ik})^2} \le R_j. \tag{2.17}$$

Для конструирования областей в пространстве изображений могут использоваться любые меры сходства, например, расстояния с весовыми коэффициентами (2.18) – (2.20), расстояние по Камберра (2.21) и т.д.

$$L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^2}, \tag{2.18}$$

$$L(S_i, X_j) = \sqrt[\lambda]{\sum_{k=1}^{n} \eta_k (s_{ik} - x_{jk})^{\lambda}}, \tag{2.19}$$

$$L(S_i, X_j) = \sum_{k=1}^{n} \eta_k \left| s_{ik} - x_{jk} \right|, \tag{2.20}$$

$$L(S_i, X_j) = \sum_{k=1}^{n} \frac{\left| s_{ik} - x_{jk} \right|}{\left| s_{ik} + x_{jk} \right|}, \tag{2.21}$$

η_k $(k = \overline{1, n})$ – весовые коэффициенты; λ – целое положительное число, большее двух.

Решающее правило (2.16) для расстояний (2.18) – (2.20) принимает вид

$$S_i \in X_j, \text{ если } L(S_i, X_j) = R_{ij} \le R_j,$$

где R_{ij} – расстояние, заданное одним из выражений (2.18) – (2.21), между предъявленным изображением S_i и центром шара, содержащего изображения j-го образа; R_j – радиус шара, содержащего изображения j-го образа.

При использовании для распознавания угла между векторами непересекающиеся области V_j $(j = \overline{1, m})$ задаются в виде конусов, а решающее правило имеет вид

$$S_i \in X_j, \text{ если } L(S_i, X_j) =$$

$$= \varphi_{ij} = \arccos\left(\frac{s_{i1}x_{j1} + s_{i2}x_{j2} + \ldots + s_{in}x_{jn}}{\sqrt{s_{i1}^2 + s_{i2}^2 + \ldots + s_{in}^2} \cdot \sqrt{x_{j1}^2 + x_{j2}^2 + \ldots + x_{jn}^2}}\right) \le \varphi_{j\max},$$

где φ_{ij} – угол между предъявленным изображением S_i и эталонным изображением X_j; $\varphi_{j\max}$ – предельно допустимый угол для j-го образа между эталонным и распознаваемым изображениями.

2.8. Распознавание объектов по качественным признакам

В большинстве случаев образы и отдельные изображения характеризуются с помощью количественных характеристик: геометрических размеров, веса, площади, объема и т. д. В этих случаях количественные изменения характеристик конкретного изображения обычно не сразу ведут к изменению образа, к которому относится распознаваемое изображение. Только достигнув опреде-

ленных для каждого образа границ, количественные изменения вызывают качественный скачок – переход к другому образу. Образы и конкретные изображения могут характеризоваться не только количественными, но и качественными характеристиками (свойствами, признаками, атрибутами). Эти признаки не могут быть описаны (или обычно не описываются) количественно, например, цвет, вкус, ощущение, запах. Образы либо обладают какими-то качественными характеристиками, либо не обладают.

Между качественными и количественными характеристиками образов есть существенное различие, однако это различие во многих случаях нельзя абсолютизировать, поскольку каждому качественному атрибуту присущи и определенные интервалы изменения количественных характеристик, за пределами которых меняется и качественный атрибут. Например, определенному цвету изображения соответствует конкретный диапазон длин электромагнитных волн, за пределами которого цвет изменится.

Существуют различные подходы к распознаванию изображений с качественными характеристиками.

В данной работе рассмотрим один из них, основанный на двоичном кодировании наличия или отсутствия какого-либо качественного признака. В рассматриваемом подходе конкретное изображение X_k некоторого образа с качественными характеристиками представляется в виде двоичного вектора

$$X_k = (x_{k1}, x_{k2}, \ldots, x_{kj}, \ldots, x_{kn}),$$

где n – размерность пространства признаков.

Если изображение X_k обладает j-м признаком, то $x_{kj} = 1$, аесли нет, то $x_{kj} = 0$, т. е. здесь отождествляется объект и двоичный вектор, его описывающий.

Рассмотрим в качестве примера четыре объекта (вишня, апельсин, яблоко, дыня), каждый из которых имеет три признака: цвет, наличие косточки или семечек (табл. 2.1).

Таблица 2.1 – Наличие свойств объектов

	Вектор признаков	Желтый цвет	Оранжевый цвет	Красный цвет	Есть косточка	Есть семечки
Вишня	X_1	Нет	нет	да	да	нет
Апельсин	X_2	Нет	да	нет	нет	да
Яблоко	X_3	Да	нет	да	нет	да
Дыня	X_4	Да	нет	нет	нет	да

В табл. 2.2 приведены числовые значения признаков для рассматриваемого примера после их двоичного кодирования.

Таблица 2.2 – Кодирование свойств объектов

	Вектор-признаков	Желтый цвет	Оранжевый цвет	Красный цвет	Есть косточка	Есть семечки
Вишня	X_1	$x_{11} = 0$	$x_{12} = 0$	$x_{13} = 1$	$x_{14} = 1$	$x_{15} = 0$
Апельсин	X_2	$x_{21} = 0$	$x_{22} = 1$	$x_{23} = 0$	$x_{24} = 0$	$x_{25} = 1$
Яблоко	X_3	$x_{31} = 1$	$x_{32} = 0$	$x_{33} = 1$	$x_{34} = 0$	$x_{35} = 1$
Дыня	X_4	$x_{41} = 1$	$x_{42} = 0$	$x_{43} = 0$	$x_{44} = 0$	$x_{45} = 1$

Наиболее простой метод решения задач распознавания объектов с качественными характеристиками после двоичного кодирования атрибутов – свести решение исходной задачи к решению задачи распознавания объектов с количественными характеристиками в n-мерном векторном пространстве. Для этого необходимо для каждого качественного признака ввести в n-мерном векторном пространстве ось. Если для рассматриваемого объекта признак существует, то на оси откладывается единица, если нет – то нуль. В результате получается многомерное двоичное пространство признаков, где можно использовать различные расстояния, применяемые для распознавания объектов с количественными характеристиками.

В рассматриваемом примере в результате введения количественных ха-

рактеристик вместо качественных признаков (табл. 2.2) получается пятимерное двоичное пространство, где можно применять расстояния по Евклиду (2.2), по Минковскому (2.3), расстояние, использующее сумму модулей разностей между соответствующими компонентами n-мерных векторов (2.4).

При двоичном кодировании качественных признаков может применяться и расстояние по Хеммингу, которое вводится для любых двоичных векторов. Расстояние по Хеммингу между двумя двоичными векторами равно числу несовпадающих двоичных компонент векторов. Если вектора имеют все одинаковые компоненты, то расстояние между ними равно нулю, если вектора не имеют совпадающих компонент, то расстояние равно размерности векторов.

Более тонкая классификация объектов с качественными признаками получается при введении для каждой пары объектов X_j, X_i, для которых введено двоичное кодирование качественных признаков, переменных, характеризующих их общность или различие с помощью табл. 2.3.

Таблица 2.3 – Взаимосвязь признаков

X_j	X_i	
	1	0
1	a	h
0	g	b

Переменная a в табл. 2.3 предназначена для подсчета числа общих признаков объектов X_j, и X_i. Она может быть вычислена с помощью соотношения

$$a = \sum_{k=1}^{n} x_{jk} x_{ik},$$

где x_{jk}, x_{ik} – двоичные компоненты векторов, описывающих объекты X_j и X_i.

С помощью переменной b подсчитывается число случаев, когда объекты X_j и X_i не обладают одним и тем же признаком,

$$b = \sum_{k=1}^{n} (1 - x_{jk})(1 - x_{ik}).$$

Переменные g и h предназначены соответственно для подсчета числа признаков, присутствующих у объекта X_i и отсутствующих у объекта X_j, и, присутствующих у объекта X_j и отсутствующими у объекта X_i,

$$g = \sum_{k=1}^{n} x_{ik}(1 - x_{jk}), \quad h = \sum_{k=1}^{n} (1 - x_{ik}) x_{jk} .$$

Из анализа переменных a, b, g, h следует, что, чем больше сходство между объектами X_j и X_i, тем больше должна быть переменная a, т.е. мера близости объектов или функция сходства должна быть возрастающей функцией от a, функция сходства должна быть симметричной относительно переменных g и h.

Относительно переменной b однозначный вывод сделать не удается, поскольку, с одной стороны, отсутствие одинаковых признаков у объектов может свидетельствовать об их сходстве, однако, с другой стороны, если у объектов общим является только отсутствие одинаковых признаков, то они не могут относиться к одному классу.

Наиболее часто применяются следующие функции сходства:

$$S_1(X_i, X_j) = \frac{a}{a + b + g + h} = \frac{a}{n} \quad \text{(функция сходства Рассела и Рао),}$$

$$S_2(X_i, X_j) = \frac{a}{n - b} \quad \text{(функция сходства Жокара и Нидмена),}$$

$$S_3(X_i, X_j) = \frac{a}{2a + g + h} \quad \text{(функция сходства Дайса),}$$

$$S_4(X_i, X_j) = \frac{a}{a + 2(g + h)}$$ (функция сходства Сокаля и Снифа),

$$S_5(X_i, X_j) = \frac{a + b}{n}$$ (функция сходства Сокаля и Мишнера),

$$S_6(X_i, X_j) = \frac{a}{g + h}$$ (функция сходства Кульжинского),

$$S_7(X_i, X_j) = \frac{ab - gh}{ab + gh}$$ (функция сходства Юла).

Предлагаем читателю самостоятельно проанализировать области существования функций.

Контрольные вопросы

1. Что такое образ?

2. Какова структура систем распознавания образов?

3. Какие есть фазы распознавания образов?

4. Как представляют образ в *n*-мерном пространстве при распознавании?

5. Какие есть меры подобия между образами в *n*-мерном пространстве?

6. Задайтесь числом *n* качественных характеристик объектов и числом *m* эталонных изображений образов (*n* и *m* должны быть не менее 4). Задайтесь несколькими объектами и с помощью функций сходства определите их принадлежность к тому или иному образу.

7. Предложите свою уникальную функцию сходства для объектов с качественными характеристиками и покажите ее работоспособность на примерах п. 6.

8. Предложите для одной из функций сходства (2.2) - (2.10) примеры распознавания, в одном из которых функция сходства должна принимать минимальное значение, а в другом – максимальное.

9. Предложите несколько примеров распознавания с помощью расстояния Хемминга. В одном из примеров расстояние по Хеммингу должно принимать значение, равное Вашему номеру по списку в журнале группы.

10. Предложите пример распознавания, в котором величина расстояния по Хеммингу будет равна величине одной из функций сходства (2.2) - (2.10).

Ученье без размышлений – пустая трата времени,
Размышления без ученья – смехотворны или гибельны.

Конфуций (551-479 гг. до н.э.)

3. ОБУЧЕНИЕ В СИСТЕМАХ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Под обучением в системах ИИ, как в психологии так и в обыденной жизни, понимают способность системы (или программы) приобретать ранее неизвестные навыки и умения. Идеи возможности обучения и самообучения компьютера возникли с первых дней их существования. Казалось, что благодаря возможности быстрого и надежного запоминания больших объемов различных сведений, самостоятельного обучения и обучения с помощью экспертов, вычислительные машины смогут быстро превзойти человека в любой области. Однако десятилетия труда кибернетиков, психологов и программистов с момента появления первых компьютеров показали, что за внешней простотой идеи обучения скрываются чрезвычайно сложные проблемы, и в настоящее время вычислительным машинам еще очень далеко в этой области до человека [1-5].

3.1. Типы обучения

Многолетний труд тысяч специалистов в области обучения искусственных систем все же не пропал даром и налицо имеется определенный прогресс в этом направлении. Прежде всего, более глубоко изучены процессы обучения

человека и установлено, что существуют различные уровни обучения, связанные между собой иерархической структурой (рис. 3.1) [4].

Уровень 1 представляет собой простое занесение в память компьютера того, что он должен “знать”. К первому уровню относятся подавляющее большинство обычных компьютерных программ современных ПК. Программа или робот при выполнении конкретных условий, заданных программистом, осуществляет то, что однозначно определено и может меняться только путем перепрограммирования. Таковы, например, действия промышленных роботов первого поколения, движения которых однозначно сформированы движениями квалифицированного рабочего, выполнявшего ту или иную технологическую операцию и одновременно программировавшего будущие действия робота, запускаемого определенным кодом и способного только слепо повторять движения человека, независимо от состояний внешней среды и объекта воздействия [4, 5].

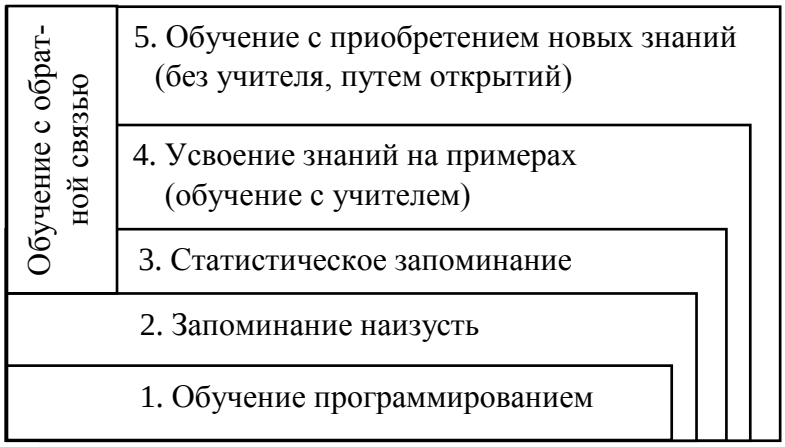


Рис. 3.1. Уровни обучения в системах ИИ

Уровень 2, или запоминание наизусть предполагает запоминание ситуаций и соответствующих им действий. Примерами интеллектуальных программ с запоминанием наизусть могут служить программы, написанные инженером фирмы IBM А.Сэмюэлем для игры в шашки на доске 8×8 в период с 1956 по 1967 гг. Как известно, шашки ходят по диагоналям вперед на соседнее свобод-

ное поле, а бьют – вперед и назад, перескакивая через шашку противника и становясь на соседнее за этой шашкой свободное поле. Дамки, в отличие от шашек, могут ходить вперед и назад на любое из свободных полей по диагонали и бить шашку противника независимо от числа свободных до нее полей, но если имеется хотя бы одно свободное поле за этой шашкой. Шашки могут превращаться в дамки, дойдя до последней горизонтали поля.

Первая программа, написанная А.Сэмюэлем в 1956 г., содержала около 180 тыс. позиций, заимствованных из лучших книг по игре в шашки. Поскольку даже для машин 2000-х годов это достаточно много, то А.Сэмюэль с целью ускорения доступа упорядочил позиции вначале по числу шашек и дамок у обоих партнеров, а затем – по частоте их появления в играх программы. При приближении памяти к переполнению использовалась процедура исключения позиций, мало встречавшихся (или совсем не возникавших) в партиях программы за какой-то последний период времени [4].

Уровень 3 – статистическое запоминание. Оно является усовершенствованным запоминанием наизусть, так как в этом случае хранится информация, полученная на основе анализа не отдельных, а большого числа случаев. Вернемся к программам игры в шашки А.Сэмюэля. Хотя 180 тысяч позиций – это и много, однако, это на порядки меньше, чем может быть при игре. Поэтому, если позиции в памяти не было, то ход машина рассчитывала с помощью полиномиальной функции оценивания

$$F = \sum_{i=1}^{38} a_i P_i, \qquad (3.1)$$

где a_i – параметры; P_i – характеристики, среди которых можно отметить контроль центра, материальный выигрыш, число угроз, число продвинутых шашек и т.д.

В первых программах расчет функции оценивания (3.1) выполнялся для глубины в три полухода (варианты, в которых на последнем полуходе была

подстановка под удар своей шашки или взятие шашки противника, продолжались до более спокойных позиций и достигали порой до двадцати полуходов).

Функция F могла использоваться и в тех позициях, где лучшие ходы были уже известны. В идеальном случае функция (3.1) должна была давать те же лучшие ходы, что записаны в памяти машины, однако это случалось далеко не всегда, и возникала необходимость корректировать параметры a_i. Коррекция выполнялась следующим образом. Программа рассчитывала по F число X лучших и число L худших ходов по сравнению с хранящимися в памяти, а затем вычисляла коэффициент

$$K = \frac{(X - L)}{(X + L)}.$$

Если $L = 0$ и, следовательно, $K = 1$, то наблюдается полное соответствие между оценками позиций по выражению (3.1) и теми оценками, которые заложены в память машины из лучших книг. Естественно, что функция F в этом случае идеальна, и менять ее не нужно. Если $X = 0$, то $K = -1$, и наблюдается полное расхождение между оценками из книг и с помощью функции (3.1). Понятно, что нельзя одним коэффициентом K оценивать все 38 параметров a_i, поэтому в программах А. Сэмюэля каждый параметр оценивался своим коэффициентом

$$K_i = \frac{(X_i - L_i)}{(X_i + L_i)},$$

где X_i и L_i – соответственно число ходов, превосходящих лучший ход из книг и уступающих ему по величине одночлена a_iP_i.

Программа, обучаясь по лучшим партиям шашечных мастеров, могла найти оптимальные параметры оценочной функции. Для определения хороших наборов коэффициентов требовалось порядка двадцати партий, при этом для полиномиального оценивания достигался 32 %-й выбор наилучшего хода. Такой относительно низкий процент лучших ходов связан с простым видом оце-

нивающего выражения (3.1).

Уровень 4 – усвоение знаний на примерах с помощью учителя. Приведенный пример получения коэффициентов оценочной функции (3.1) при игре в шашки можно рассматривать и как обучение с учителем, в результате которого происходит статистическое запоминание. В обучении с учителем есть одна опасность: если учитель плох, то и обучающаяся программа или система будет соответствующего качества. Это обнаружил и А.Сэмюэль – если программа обучалась в игре со слабыми партнерами, то качество ее игры резко падало.

Уровень 5 – обучение без учителя. Этот вид обучения также можно иллюстрировать примером из шашек. Программа может играть сама против себя, пользуясь разными вариантами функции оценивания F и извлекая знания самостоятельно. Пусть A – лучшая версия программы, оценочную функцию которой обозначим F_a. Пусть эта программа играет против программы B с оценочной функцией F_b. Если выигрывает программа A, то слегка меняются параметры функции F_b и играется следующая партия. Если выигрывает программа B, то программа A получает штрафное очко. После определенной суммы штрафных очков производится взаимный обмен функциями F_a и F_b и существенно меняются параметры F_b неудачно игравшей программы.

Иногда третий, четвертый и пятый уровни обучения объединяют под общим названием обучение *с обратной связью*.

3.2. Структура обучающейся системы

Пример структуры обучающейся системы приведен на рис. 3.2. Для простоты рассматривается система, которая обучается различать только два объекта, выделяя при этом их общие свойства. Система работает с объектами, которые характеризуются *именем* и совокупностью *атрибутов*. Атрибуты, в свою очередь, содержат некоторую *информацию*, представленную как в числовой, так и в логической форме. Кроме этого, атрибуты имеют свой тип, который задается *индексом* (i).

Система может работать в трех режимах: обучение с учителем, самообучение и распознавание.

В начале работы задается априорная информация о распознаваемых объектах системы (вводятся имена и атрибуты объектов). После этого с консоли или другого устройства вводятся атрибуты неизвестного объекта.

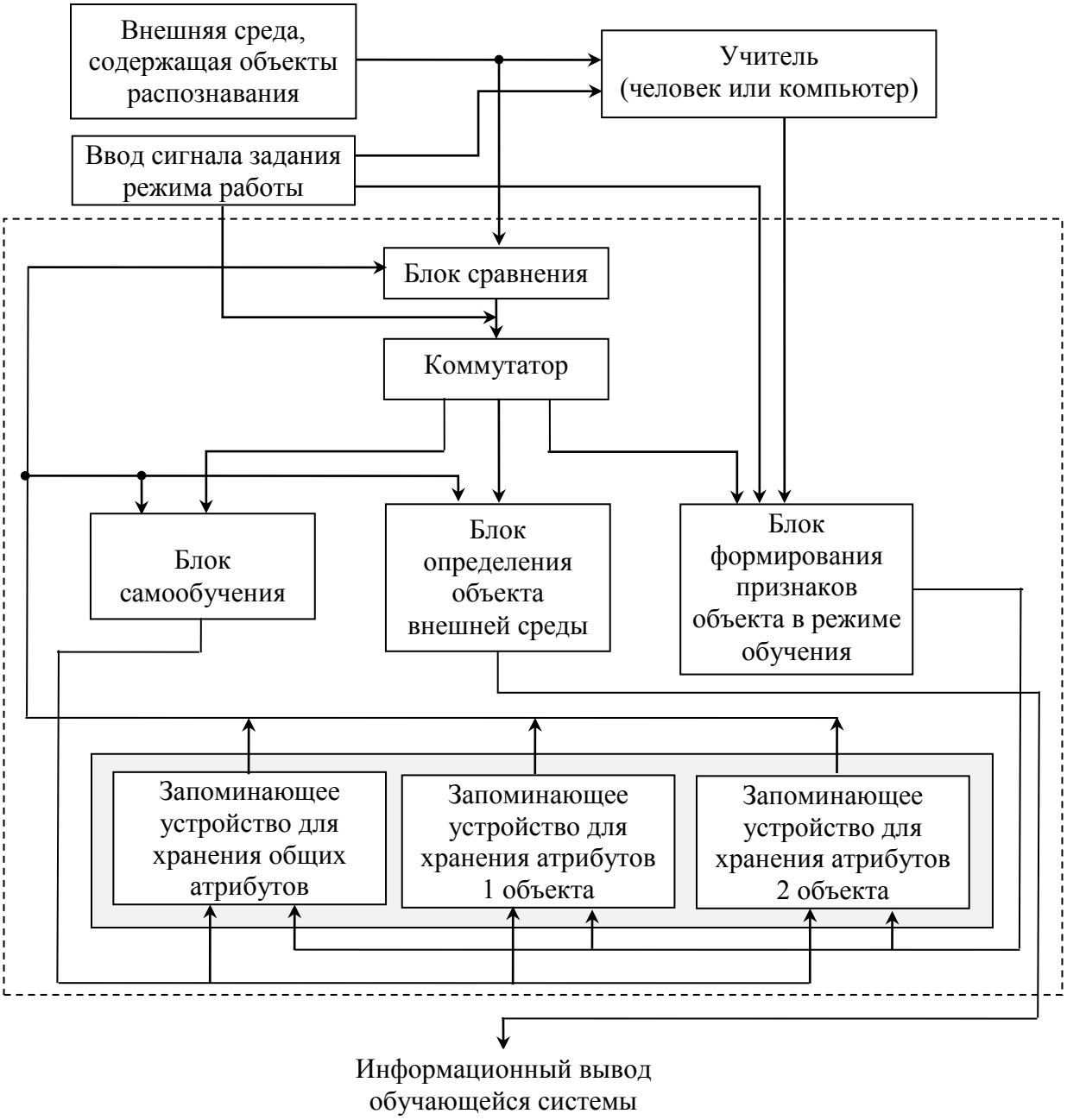


Рис. 3.2. Структура обучающейся системы

В *режиме распознавания* система соотносит введенные атрибуты с атрибутами известных объектов и определяет, к какому из них можно отнести неизвестный объект.

В *режиме обучения с учителем* учитель дает указания системе, к какому из объектов (1 или 2) отнести введенные атрибуты (т.е. сам распознает объект).

В *режиме самообучения* система сама соотносит введенные атрибуты с одним из объектов.

После опознания объекта в режиме обучения система производит модификацию атрибутов этого объекта в зависимости от индекса атрибута.

Индекс определяет способ обработки атрибута в режиме обучения. Возможны следующие способы обработки.

1. Запоминание атрибутов об объекте. Атрибут, как он есть в чистом виде, запоминается в списке атрибутов объекта.

2. Запоминание атрибута с выделением списка общих атрибутов. Кроме запоминания в списке атрибутов объекта, проверяется наличие такого же атрибута у другого объекта, и если он есть, то выносится в список общих атрибутов.

3. Запоминание атрибутов и статистическая обработка (модификация числовой характеристики атрибута).

4. Запоминание атрибутов, статистическая обработка и выделение списка общих атрибутов.

Для атрибутов вводится также и способ статистической обработки. Она применяется для тех атрибутов, которые имеют числовую характеристику, для логических атрибутов отмечается, что статистическая обработка отсутствует ($i = 0$).

3.3. Алгоритмы обучения

Возможны различные типы статистической обработки, используемой для обучения, основанные на вычислении среднего значения числовой характеристики атрибута при его многократном появлении и уточнении интервала значе-

ний:

1) Вычисление среднего арифметического (i = 1) после предъявления атрибута a_i в (n + 1)-й раз выполняется по соотношению

$$S_{ai}(n+1) = \frac{(n(S_{ai}(n) + a_i(n+1))}{n+1}, \tag{3.2}$$

где

$$S_{ai}(n) = \frac{(a_i(1) + a_i(2) + \ldots + a_i(n)}{n} \tag{3.3}$$

среднее арифметическое после n предъявлений атрибута a_i.

2) Вычисление среднего с весом q (i = 2) после предъявления атрибута a_i в (n + 1)-й раз выполняется по формуле

$$S_{qai}(n+1) = (1 - q_{n+1})S_{qai}(n) + q_{n+1}a_i(n+1)),$$

где $S_{qai}(n+1), S_{qai}(n)$ соответственно среднее с весом после предъявления атрибута a_i в (n + 1) и n-й раз.

Полагая, что последние предъявления наиболее информативны, например в силу того, что атрибут может меняться в процессе обучения, зададим, что при (n + 1)-м предъявлении атрибута вес q_{n+1} определяется выражением

$$q_{n+1} = \begin{cases} 1, \text{если } n = 1, \\ \dfrac{1}{n}, n = 2, 3, \ldots, m, \\ \dfrac{1}{m+1}, \text{если } n \geq m+1, \end{cases}$$

а веса $q_1, q_2, \ldots, q_n$ равны между собой и их сумма определяется следующим выражением

$$\sum_{j=1}^{n} q_j = 1 - q_{n+1}.$$

3) Вычисление среднего геометрического (i = 3) после предъявления атрибута в (n + 1)-й раз выполняется по соотношению

$$S_{gai}(n+1) = ((S_{gai}(n))^n a_i(n+1))^{\frac{1}{n+1}},$$

где $S_{gai}(n)$ среднее геометрическое, после предъявлений атрибута вычисляется по соотношению

$$S_{ai}(n) = (a_i(1) \times a_i(2) \times \ldots \times a_i(n))^{\frac{1}{n}}.$$

4) Определение интервала, в котором должны находиться числовые характеристики атрибута при его многократном появлении (i = 4). При этом определяется интервал [$a_{i\min}(m)$, $a_{i\max}(m)$], в котором должны находиться числовые характеристики i-го атрибута после его m предъявлений. При первом появлении i-го атрибута $a_i(1)$ задают $a_{i\min}(1) = a_{i\max}(1) = a_i(1)$. При следующем появлении атрибута, когда выполняется неравенство $a_i(k) \neq a_i(1)$, $k \geq 2$, определяют граничные точки интервала соотношениями:

$$a_{i\min}(k) = a_i(1),\ a_{i\max}(k) = a_i(k),\ \text{если}\ a_i(k) > a_i(1);$$
$$a_{i\min}(k) = a_i(k),\ a_{i\max}(k) = a_i(1),\ \text{если}\ a_i(k) < a_i(1).$$

При последующих появлениях i-го атрибута граничные точки интервала задаются соотношениями

$$a_{i\min}(n+1) = a_{i\min x}(n),\ a_{i\max}(n+1) = a_{i\max}(n),\ \text{если}\ a_{i\min}(n) \le a_i(n+1) \le a_{i\max}(n);$$
$$a_{i\min}(n+1) = a_i(n+1),\ a_{i\max}(n+1) = a_{i\max}(n),\ \text{если}\ a_n(n+1) < a_{i\min}(n);$$
$$a_{i\min}(n+1) = a_{i\min}(n),\ a_{i\max}(n+1) = a_i(n+1),\ \text{если}\ a_i(n+1) > a_{i\max}(n).$$

Как следует из общей структурной схемы интеллектуальной системы, в ее основу, в режиме обучения заложены механизмы запоминания фактов, классифицируемых учителем, их обработка (статистическая, если она предполагается, и (или) выделение общих атрибутов двух объектов) и, быть может, механизмы логического вывода, вскрывающие противоречия между вновь предъявленными и хранящимися в памяти фактами.

В режимах распознавания и распознавания с самообучением должны работать механизмы логического вывода, на основе которых принимается одно из следующих решений:

- ✓ предъявлен первый объект;
- ✓ предъявлен первый объект, но имеются неизвестные атрибуты;
- ✓ предъявлен второй объект;
- ✓ предъявлен второй объект, но имеются неизвестные атрибуты;
- ✓ распознавание невозможно, так как предъявлены только общие атрибуты двух объектов; необходима дополнительная информация;
- ✓ распознавание невозможно, так как предъявлены только атрибуты, отсутствующие в памяти системы; необходимо дополнительное обучение;
- ✓ распознавание невозможно, так как предъявлены атрибуты, двух разных объектов.

В режиме распознавания с самообучением, когда принимается решение – предъявлен *k*-й (k = 1, 2) объект, но имеются неизвестные атрибуты, должен быть механизм записи неизвестных атрибутов в ЗУ *k*-го объекта.

Из приведенного выше следует, что система обучается только в том случае, если ей предъявляется новая информация, которая отсутствует в памяти или противоречит уже имеющейся.

3.4. Пример работы обучающейся системы

Рассмотрим обучающуюся программу, которая должна распознавать два объекта: ПК конфигурации 1 на базе процессора Intel и ПК конфигурации 2 на базе процессора AMD. Некоторые общие свойства и свойства, присущие только каждому из этих двух объектов, приведены на рис. 3.3.

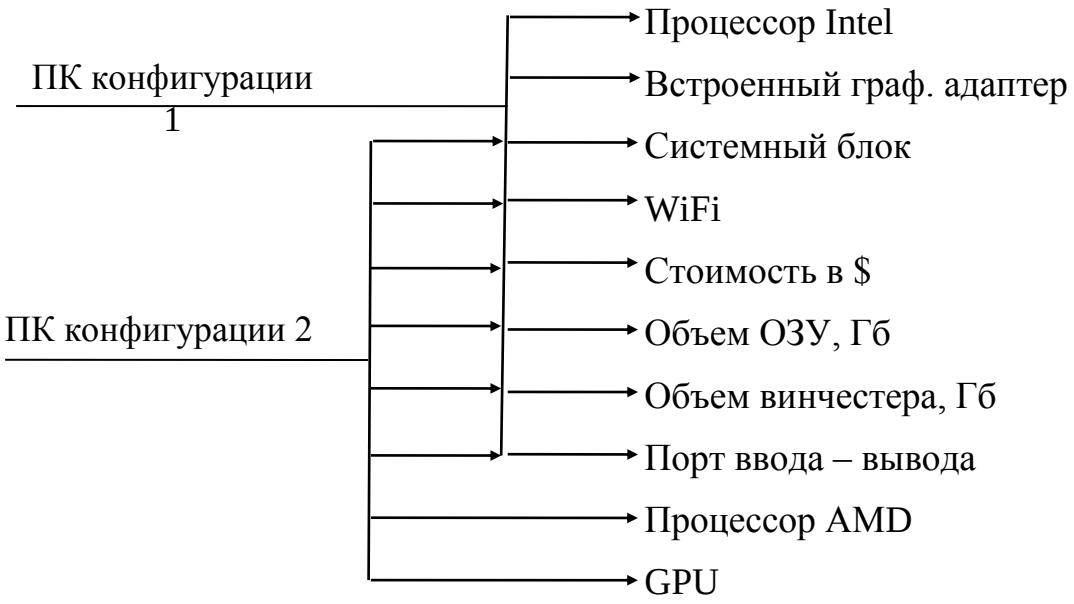


Рис. 3.3. Свойства распознаваемых объектов

Часть общих свойств (стоимость, объем винчестера и оперативного ЗУ), хотя и присущи обоим объектам, но существенно отличаются своими числовыми характеристиками, по величине которых легко распознать предъявляемый объект. В режиме обучения числовые значения этих свойств могут изменяться (например, могут предъявляться компьютеры с разными объемами ОЗУ и винчестера, разной стоимостью), что вызывает необходимость статистической обработки входной информации. Рассмотрим несколько простейших способов такой обработки.

Будем рассматривать обучение системы (программы) на примере двух конкретных объектов, но закладывать в его основу общие принципы распознавания двух любых объектов, а также предполагать возможность распростране-

ния этих принципов на распознавание трех и большего числа объектов.

Для обеспечения работы программ с произвольной парой объектов необходимо, чтобы она в режиме обучения, прежде всего, запрашивала имена распознаваемых объектов:

Имя 1-го объекта? ПК1 Имя 2-го объекта? ПК2

Затем должен быть запрос об общем характере обработки атрибутов объектов. Предположим, что в программе возможны три способа обработки входной информации:

✓ запоминание атрибутов объектов;

✓ запоминание атрибутов объектов с выделением списка общих атрибутов;

✓ запоминание атрибутов объектов с выделением списка общих атрибутов и статистической обработкой числовой информации атрибутов.

В этом случае на запрос программы возможны три разных ответа:

Характер обработки атрибутов? Запоминание Характер обработки атрибутов? Запоминание и выделение общих атрибутов. Характер обработки атрибутов? Запоминание и выделение общих атрибутов, статистическая обработка.

При наличии в программе режима записи априорной информации делается запрос об использовании этого режима:

Запись априорной информации? Да/Нет.

Если ответ “Да”, то программа приступает к вводу атрибутов первого объекта. В связи с тем, что числовые значения атрибутов могут подвергаться различным видам статистической обработки, то, кроме имен атрибутов и их числовых характеристик, необходимо вводить еще и числовое значение переменной, обозначенной через *i*, указывающей вид необходимой обработки по-

ступающих данных (соответственно i = 1, 2, 3 или 4) или отсутствие такой обработки (i = 0):

Введите атрибуты первого объекта и Enter и конце Введите атрибут 1? i = 0, Встроен. граф. адаптер Введите атрибут 2? i = 1, стоимость, 500 $ Введите атрибут 3? i = 4, объем винчестера, 1000Гб Введите атрибут 4? i = 0, процессор Intel Введите атрибут 5? Enter

Затем идет ввод атрибутов второго объекта и атрибутов общего списка:

Введите атрибуты второго объекта и Enter и конце Введите атрибут 1? i = 0, процессор AMD Введите атрибут 2? i = 1, стоимость, 600 $ Введите атрибут 3? i = 4, объем винчестера, 1500 Гб Введите атрибут 4? Enter
Введите атрибуты общего списка и Enter и конце Введите атрибут 1? i = 0, Wi-Fi Введите атрибут 2? i = 0, системный блок Введите атрибут 3? Enter

После этого на экран выводятся списки атрибутов обоих объектов:

Список 1 (ПК1)	Список 2 (ПК2)	Список 3 (общий)
Встроен. граф. адаптер стоимость 500 $ объем винчестера 1000 Гб	процессор AMD стоимость 600 $ объем винчестера 1500 Гб	Wi-Fi системный блок

Затем программа задает вопрос о дальнейшем режиме работы:

Режим работы? 1 – запись априорной информации; 2 – режим обучения с учителем; 3 – режим распознавания; 4 – режим распознавания с самообучением; 5 – конец работы с программой.

Пусть выбран режим работы – обучение с учителем 2.

После этого ответа программа запросит ввод атрибутов любого объекта:

Введите атрибуты любого объекта Введите атрибут 1? *i* = 1, стоимость, 530 $ Введите атрибут 2? *i* = 0, Wi-Fi Введите атрибут 3? *i* = 0, порт ввода-вывода Введите атрибут 4? *i* = 4, объем винчестера, 750Гб Введите атрибут 5? Enter

Получив атрибуты неизвестного объекта, программа вычисляет оценочные функции F_1 и F_2 полиномиального вида для каждого из объектов:

$$F_1 = \sum_{j=1}^{k} P_j^1,\ F_2 = \sum_{j=1}^{k} P_j^2,$$

$$P_j^q = \begin{cases} 1, \text{ если } j\text{-й атрибут имеется во введенном списке} \\ \quad \text{и в памяти атрибутов } q\text{-го объекта, или} \\ \quad \text{если } i = 1, 2, 3 \text{ и } \left|a_j - S_j^{iq}\right| \le \left|a_j - S_j^{ik}\right|,\ q, k = 1, 2,\ q \ne k, \text{ или} \\ \quad \text{если } i = 4 \text{ и } a_{j\min}^q \le a_j \le a_{j\max}^q, \\ 0, \text{ если } i = 0 \text{ и } j\text{-й атрибут отсутствует} \\ \quad \text{в памяти атрибутов } q\text{-го объекта, или} \\ \quad \text{если } i = 1, 2, 3 \text{ и } \left|a_j - S_j^{iq}\right| > \left|a_j - S_j^{ik}\right|,\ q, k = 1, 2,\ q \ne k, \text{ или} \\ \quad \text{если } i = 4 \text{ и } a_{j\min}^q > a_j \text{ или } a_j > a_{j\max}^q, \end{cases}$$

где k – число атрибутов, введенных программой на предыдущем этапе работы; a_j – численное значение j-го атрибута во введенном списке; S_j^{iq} – среднее значениеj-го атрибута в памяти атрибутов сравниваемого q-го (q = 1, 2) объекта при i-м способе статистической обработки входных данных; $a_{j\min}^q, a_{j\max}^q$ – граничные точки допустимого интервала изменений j-го атрибута для q-го объекта при четвертом способе статистической обработки числовых данных.

После вычисления функций F_1 и F_2 производится логический вывод:

- ✓ предъявлен 1-й объект, если $F_1 > F_2$;
- ✓ предъявлен 2-й объект, если $F_1 < F_2$;
- ✓ предъявлен k-й (случайно выбранный) объект, если $F_1 = F_2$.

В рассматриваемом примере в соответствии с последними выражениями имеем:

$$F_1 = \sum_{j=1}^{4} P_j^1 = 1+1+0+1 = 3,$$

$$F_2 = \sum_{j=1}^{4} P_j^2 = 0+1+0+0 = 1.$$

Отсюда следует, что в этом случае выполняется условие, что предъявлен первый объект, поэтому программа запрашивает учителя:

Это ПК1? Да.

Затем программа обрабатывает списки атрибутов с учетом новой информации и выводит их на экран:

Список 1 (ПК1)	Список 2 (ПК2)	Список 3 (общий)
Встроен. граф. адаптер стоимость 515 $ объем винчестера [750,1000] Гб порт ввода-вывода	процессор AMD стоимость 600 $ объем винчестера 1500 Гб	Wi-Fi системный блок

Затем задается вопрос о дальнейшем режиме работы:

Режим обучения продолжается? Да Введите атрибуты любого объекта Введите атрибут 1? i = 0, Wi-Fi Введите атрибут 2? i = 0, системный блок Введите атрибут 3? i = 0, порт ввода-вывода Введите атрибут 4? i = 0, процессор AMD Введите атрибут 5? i = 4, объем ОЗУ4 Гб Введите атрибут 6? Enter

В рассматриваемом случае оценочные функции F_1 и F_2совпадают по величине, поэтому программа должна сделать случайный выбор объекта.

Пусть в данном примере это будет первый объект. После этого программа запрашивает учителя:

Это ПК1? Нет.

Вслед за этим программа с учетом новой информации корректирует списки атрибутов и выводит их на экран:

Список 1 (ПК1)	Список 2 (ПК2)	Список 3 (общий)
Встроен. граф. адаптер стоимость 515 $ объем винчестера [750,1000] Гб	процессор AMD стоимость 600 $ объем винчестера 1500 Гб объем ОЗУ4 Гб	Wi-Fi системный блок порт ввода – вывода

Далее задается вопрос о дальнейшем режиме работы:

Режим обучения продолжается? Нет Режим работы? 1 – запись априорной информации; 2 – режим обучения с учителем; 3 – режим распознавания; 4 – режим распознавания с самообучением; 5 – конец работы с программой.

Пусть следующим режимом работы будет режим распознавания с самообучением:

4 Введите атрибуты распознаваемого объекта Введите атрибут 1? Процессор AMD Введите атрибут 2? Системный блок Введите атрибут 3? Порт ввода – вывода Введите атрибут 4? Отдельный GPU Введите атрибут 5? Enter

Вычисление оценочных функций F_1 и F_2 для предъявленного объекта дает, что $F_2 > F_1$, т.е. предъявлен второй объект. Однако при этом имеется атрибут (Отдельный GPU), которого нет ни в одном из списков, поэтому программа этот атрибут занесет в список второго объекта и выдаст сообщение:

Предъявлен объект ПК2, неизвестный атрибут (Отдельный GPU) внесен в список его атрибутов
Режим распознавания с самообучением продолжается? Да
Введите атрибуты любого объекта
Введите атрибут 1? Wi-Fi
Введите атрибут 2? Объем ОЗУ 6 Гб
Введите атрибут 3? Порт ввода-вывода
Введите атрибут 4? Отдельный GPU
Введите атрибут 5? Enter

Вычисление оценочных функций F_1 и F_2 для предъявленного объекта дает, что $F_2 > F_1 = 0$, т.е. предъявлен второй объект, однако при этом числовой атрибут (объем ОЗУ 6 Гб) находится вне интервала [4, 4], полученного в режиме обучения, поэтому программа расширит интервал значений атрибута “объем ОЗУ” и выдаст сообщение:

Предъявлен объект ПК2, изменен интервал атрибута объем ОЗУ с [4, 4] Гб на [4, 6] Гб
Режим распознавания с самообучением продолжается? Нет
Режим работы?
1 – запись априорной информации;
2 – режим обучения с учителем;
3 – режим распознавания;
4 – режим распознавания с самообучением;
5 – конец работы с программой.
5

Программа работу закончила.

Контрольные вопросы

1. Чем системы искусственного интеллекта отличаются от других систем?

2. Какие есть уровни обучения в системах ИИ?

3. Какова структура обучающейся системы?

4. Для чего создается список общих атрибутов объектов?

5. В чем отличие работы системы в режиме обучения с учителем от работы в режиме самообучения?

6. Какие виды статистической обработки применяются при обучении?

7. Чем отличается статистическая обработка первого вида от статистической обработки второго вида?

8. Приведите формулу для вычисления среднего геометрического множества численных значений *i*-го атрибута при его *n*-кратном появлении.

9. Когда оправдано применять интервальный вид атрибута объекта?

10. Для чего используется параметр *i* при описании атрибутов объекта?

11. На основании чего принимается решение о классе распознаваемого объекта?

12. Для чего применяется таблица общих свойств?

13. Когда корректируется таблица общих свойств?

Единственный способ улучшить наши рассуждения – это сделать их столь же осязаемыми, как математические формулы.

Лейбниц, “Искусство открытия”

4. МЕТОДЫ, ОСНОВАННЫЕ НА ЗНАНИЯХ

4.1. Введение в экспертные системы

В начале восьмидесятых годов двадцатого века в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название экспертные системы (ЭС). Цель исследований по ЭС состоит в разработке программ, которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. Программные средства (ПС), базирующиеся на технологии экспертных систем, или инженерии знаний получили значительное распространение. Технология экспертных систем существенно расширяет круг практически значимых задач, решаемых на компьютерах, решение которых приносит значительный экономический эффект.

Технология ЭС является важнейшим средством в решении глобальных проблем традиционного программирования. Длительность и, следовательно, высокая стоимость разработки сложных приложений; высокая стоимость сопровождения сложных систем, которая часто в несколько раз превосходит стоимость их разработки; низкий уровень повторной используемости программ – это обусловило быстрое развитие ЭС.

Объединение технологии ЭС с технологией традиционного программирования добавляет новые качества к программным продуктам за счет обеспечения динамичной модификации приложений пользователем, а не программистом;

большей прозрачности приложения; лучшей графики, интерфейса и т.п.

ЭС предназначены для решения неформализованных задач, т.е. ЭС не отвергают и не заменяют традиционного подхода к разработке программ, ориентированного на решение не формализованных задач, которые обладают одной или несколькими из следующих характеристик:

- задачи не могут быть заданы в числовой форме;
- цели не могут быть выражены в терминах точно определенной целевой функции;
- не существует алгоритмического решения задач.

Неформализованные задачи обычно обладают:

– ошибочностью, неоднозначностью, неполнотой и противоречивостью исходных данных;

– ошибочностью, неоднозначностью, неполнотой и противоречивостью знаний о проблемной области и решаемой задаче;

– большой размерностью пространства решения, т.е. перебор при поиске решения весьма велик;

– динамически изменяющимися данными и знаниями.

Следует подчеркнуть, что неформализованные задачи представляют большой и очень важный класс задач.

Экспертные системы и системы искусственного интеллекта отличаются от систем обработки данных тем, что в них, в основном, используются символьный (а не числовой) способ представления, символьный вывод и эвристический поиск решения, а неисполнение известного алгоритма.

По качеству и эффективности решения экспертные системы не уступают решениям эксперта-человека. Решения экспертных систем обладают прозрачностью, т.е. могут быть объяснены пользователю на качественном уровне. Это качество экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях. Экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом.

Технология экспертных систем используется для решения различных типов задач: интерпретации, предсказания, диагностики, планирования, конструирования, контроля, отладки, инструктажа и управления.

4.1.1. Структура экспертных систем.

ЭС состоит из следующих основных компонентов: решателя (интерпретатора); рабочей памяти (РП), называемой также базой данных (БД); базы знаний (БЗ); компонентов приобретения знаний; объяснительного компонента; диалогового компонента (рис. 4.1) [5-12].

База данных (*рабочая память*) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. Этот термин совпадает по названию, но не по смыслу с термином, используемым в информационно-поисковых системах (ИПС) и системах управления базами данных (СУБД) для обозначения всех данных (в первую очередь долгосрочных), хранимых в системе.

База знаний (БЗ) в ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

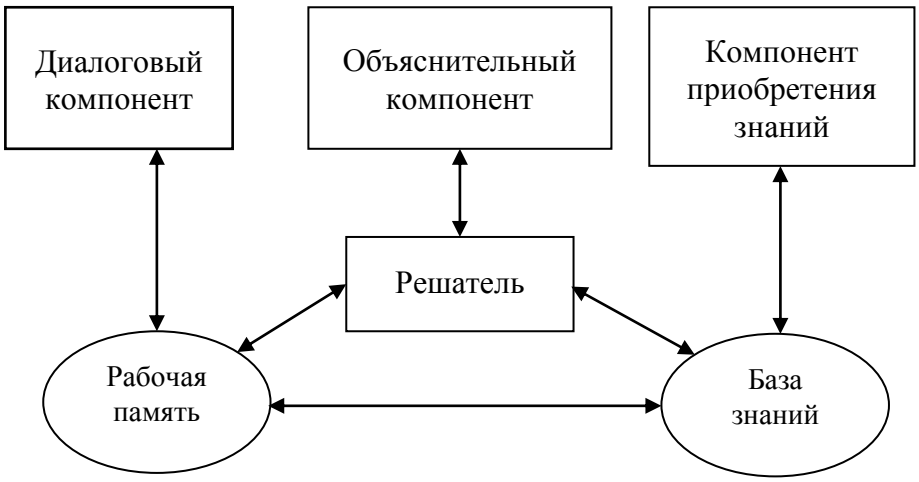


Рис.4.1. Основные компоненты ЭС

Решатель, используя исходные данные из рабочей памяти и знания из БЗ, формирует такую последовательность правил, которая приводит к решению задачи.

Компонент приобретения знаний автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

Объяснительный компонент объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.

Диалоговый компонент ориентирован на организацию дружественного общения с пользователем, как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

В разработке ЭС участвуют представители следующих специальностей:

✓ эксперт проблемной области, задачи которой будет решать ЭС;

✓ инженер по знаниям – специалист по разработке ЭС (используемые им технологии, методы называют технологией инженерии знаний);

✓ программист по разработке инструментальных средств, предназначенных для ускорения разработки ЭС.

Необходимо отметить, что отсутствие среди участников разработки инженеров по знаниям (т. е. их замена программистами) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того инструментального средства, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом средстве; выделяет и программиру-

ет (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

Программист разрабатывает инструментальное средство (если оно разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой они будут использованы.

Экспертная система работает в двух режимах: режиме приобретения знаний и в режиме решения задачи (называемом также режимом консультации или режимом использования ЭС).

В режиме приобретения знаний общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерными для рассматриваемой области.

Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт (с помощью ЭС), не владеющий программированием.

В режиме консультации общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам), или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС ру-

тинную работу). В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Решатель на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может потребовать объяснения: почему система задает тот или иной вопрос? Как ответ получен системой?

4.1.2. Этапы разработки экспертных систем.

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату.

Использовать ЭС следует только тогда, когда разработка ЭС *возможна, оправдана и* методы инженерии знаний *соответствуют* решаемой задаче. Чтобы разработка ЭС была *возможной* для данного приложения, необходимо одновременное выполнение по крайней мере следующих требований:

1) существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;

2) эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;

3) эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут извлечены и вложены в ЭС;

4) решение задачи требует только рассуждений, а не действий;

5) задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);

6) задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно понятной и структурированной области, т.е.

должны быть выделены основные понятия, отношения и известны (хотя бы эксперту) способы получения решения задачи;

7) решение задачи не должно в значительной степени использовать здравый смысл (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), так как подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Использование ЭС в данном приложении может быть возможно, но не оправдано. Применение ЭС может быть *оправдано* одним из следующих факторов:

– решение задачи принесет значительный эффект, например экономический;

– использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;

– использование ЭС целесообразно в тех случаях, когда при передаче информации эксперту происходит недопустимая потеря времени или информации.

Приложение *соответствует* методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

1) задача может быть естественным образом решена посредством символических рассуждений, а не манипуляций с числами, как принято в математических методах и в традиционном программировании;

2) задача должна иметь эвристическую, а не алгоритмическую природу, т.е. ее решение должно требовать применения эвристических правил. Задачи, которые могут быть гарантированно решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур, не подходят для применения ЭС;

3) задача должна быть достаточно сложна, чтобы оправдать затраты на разработку ЭС. Однако она не должна быть чрезмерно сложной (решение занимает у эксперта часы, а не недели), чтобы ЭС могла ее решать;

4) задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

При разработке ЭС, как правило, используется концепция быстрого прототипа. Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип ЭС. Прототипы должны удовлетворять двум противоречивым требованиям: с одной стороны, они должны решать типичные задачи конкретного приложения, а с другой – время и трудоемкость их разработки должны быть весьма незначительны, чтобы можно было максимально запараллелить процесс накопления и отладки знаний с процессом выбора программных средств. Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения.

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть следующих этапов (рис. 4.2): идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию.

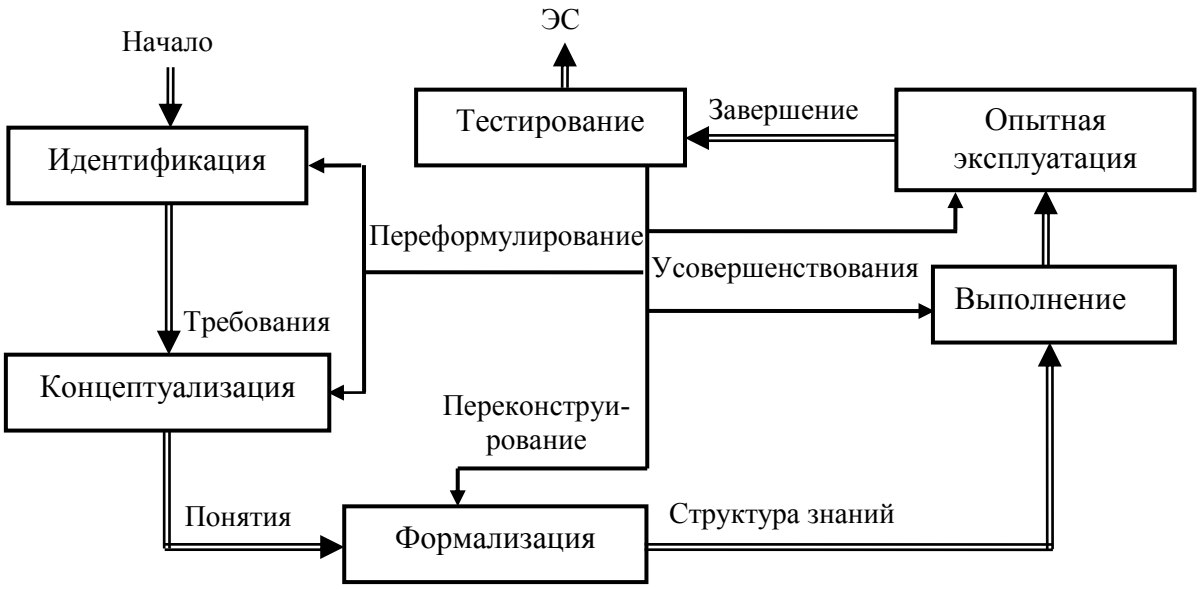


Рис. 4.2. Технология разработки ЭС

На этапе *идентификации* определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.

Во время *концептуализации* проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

В ходе *формализации* выбираются ИС и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

На этапе *выполнения* осуществляется наполнение базы знаний, создание прототипа ЭС. Главное в создании прототипа заключается в том, чтобы этот прототип обеспечил проверку адекватности идей, методов и способов представления знаний решаемым задачам. Создание первого прототипа должно подтвердить, что выбранные методы решений и способы представления пригодны для успешного решения, по крайней мере, ряда задач из актуальной предметной области, а также продемонстрировать тенденцию к получению высококачественных и эффективных решений для всех задач предметной области по мере увеличения объема знаний.

В ходе этапа *тестирования* производится оценка выбранного способа представления знаний в ЭС в целом. Для этого инженер по знаниям подбирает примеры, обеспечивающие проверку всех возможностей новой ЭС.

Различают следующие источники неудач в работе системы: тестовые примеры, ввод-вывод, правила вывода, управляющие стратегии.

Показательные тестовые примеры являются наиболее очевидной причиной неудачной работы ЭС. Поэтому при подготовке тестовых примеров следует классифицировать их по подпроблемам предметной области, выделяя стандартные случаи, определяя границы трудных ситуаций и т.п.

Критерии оценки ЭС зависят от точки зрения. При тестировании промышленной системы превалирует точка зрения инженера по знаниям, которого в первую очередь интересует вопрос оптимизации представления и манипулирования знаниями. И, наконец, при тестировании ЭС после опытной эксплуатации оценка производится с точки зрения пользователя, заинтересованного в удобстве работы и получения практической пользы.

На этапе *опытной эксплуатации* проверяется пригодность ЭС для конечного пользователя. Пригодность ЭС для пользователя определяется в основном удобством работы с ней и ее полезностью. Под полезностью ЭС понимается ее способность в ходе диалога определять потребности пользователя, выявлять и устранять причины неудач в работе, а также решать поставленные задачи. В свою очередь, удобство работы с ЭС подразумевает естественность взаимодействия с ней (общение в привычном пользователю виде), гибкость ЭС (способность системы настраиваться на различных пользователей, а также учитывать изменения в квалификации в одной и той же группе пользователей и устойчивость системы к ошибкам).

В ходе разработки ЭС почти всегда осуществляется ее модификация: переформулирование понятий и требований или переконструирование представления знаний в системе и усовершенствование прототипа.

4.1.3.Представление знаний в экспертных системах.

Круг вопросов, решаемых при представлении знаний, включает: определение состава представляемых знаний; организацию знаний; представление знаний, т.е. определение модели представления.

Состав знаний ЭС определяется следующими факторами: проблемной средой; архитектурой экспертной системы; потребностями и целями пользователей; языком общения. В соответствии с общей схемой экспертной системы (рис. 4.1) для ее функционирования требуются следующие знания:

- ✓ знания о процессе решения задачи, т.е. управляющие знания, ис-

пользуемые интерпретатором (решателем);

✓ знания о языке общения и способах организации диалога, используемые лингвистическим процессором (диалоговым компонентом);

✓ знания о способах представления и модификации знаний, используемые компонентом приобретения знаний;

✓ поддерживающие структурные и управляющие знания, используемые объяснительным компонентом.

Для динамической ЭС, кроме того, необходимы следующие знания:

1) знания о методах взаимодействия с внешним окружением;

2) знания о модели внешнего мира.

Состав знаний зависит от требований пользователя и задач (из общего набора задач), которые решаются; способов и методов решения этих задач; при каких ограничениях на результаты и способы их получения должна быть решена задача, требований к языку общения; степени конкретности знаний о проблемной области, доступности пользователям; целей пользователей.

Состав знаний о языке общения зависит как от языка общения, так и от требуемого уровня понимания.

4.2. Формальные основы экспертных систем

4.2.1.Основы исчисления предикатов.

Слово *логика* означает систематический метод рассуждений. Рассмотрим две конкретные системы логики – базисную – исчисление высказываний и более богатую – исчисление предикатов. Исчисление высказываний – совокупность правил, используемых для определения истинности или ложности некоторой комбинации высказываний. В логике разработаны определенные формализмы для представления фактов и правил.

Можно рассматривать предложения или высказывания обыденной раз-

говорной речи, и с помощью исчисления производить элементарные рассуждения.

Высказывание, или предложение, или факт – это просто утверждение, которое может быть истинно или ложно. Примеры предложений:

ЕСЛИ *Смит является специалистом по ЭВМ,* ТО *Смит – оптимист.*
СМИТ является специалистом по ЭВМ.

Из этих правил можно сделать вывод

Смит – оптимист.

Более формально выразим это, применив запись:

ЕСЛИ *P* то *Q*, *P* СЛЕДОВАТЕЛЬНО *Q*,

где *P* и *Q* предыдущие правила. Обнаружив совпадение, человек может утверждать, что суждение имеет некоторую приемлемую логическую структуру, и следовательно, может сделать вывод о том, что заключение

Смит – оптимист

справедливо. Полностью утверждение выглядит следующим образом:

ЕСЛИ *Смит является специалистом по ЭВМ,* ТО *Смит – оптимист.*
СМИТ *является специалистом по ЭВМ.*
СЛЕДОВАТЕЛЬНО
Смит – оптимист.

На обоснованности примера никак не сказывается, имеет ли содержание утверждения какой-либо смысл в реальном мире. Например, следующее утверждение справедливо, хотя и абсурдно:

ЕСЛИ *Смит является специалистом по ЭВМ*, ТО *Смит – спит.*

СМИТ *является специалистом по ЭВМ.*

СЛЕДОВАТЕЛЬНО

Смит – спит.

Факты можно переписать в виде предикатов:

является (*смит, специалист по ЭВМ*).

является (*смит, оптимист*).

является (*смит, спящий*).

Эти факты выражают единичные отношения, указанные слева от скобок, а перечисленные в скобках объекты, связаны данными соотношениями.

В исчислении предикатов именам отношений соответствует термин предикаты, а объектам – аргументы. Порядок аргументов должен всегда задаваться в соответствии с интерпретацией предиката, принятой в рамках предметной области. Предикат может иметь произвольное число аргументов. Отдельные высказывания могут объединятся в сложные высказывания с помощью логических отношений И (and, &), ИЛИ (or, V), НЕ(not, ~) и импликация (→). Использованный выше метод вывода носит латинское название *modus ponens* (сокращение утверждения). Его можно записать следующим образом:

ЕСЛИ *истинно A и истинно A → B*, ТО *можно вывести B*.

Знак → импликация (произносится влечет) применяется для формирования правил и читается “если ...., то .....”. Таким образом, из двух данных предложений можно вывести третье.

Некоторый объект может быть представлен как константа, т.е. как конкретный индивидуум, или как переменная, например:

является (*X, специалист по ЭВМ*).

Когда переменной ставится в соответствие определенное имя некоторого объекта, т.е. некоторой константы, происходит порождение экземпляра этой переменной. Здесь все константы начинаются со строчной буквы. Для того чтобы в исчислении предикатов можно было манипулировать переменными, потребовалось ввести дополнительную структуру– квантор.

Кванторы служат для указания меры, в какой экземпляры переменных должны быть истинными, для того чтобы в целом высказывание стало истинным. Различают квантор общности, обозначаемый символом $\forall$, который означает "для каждого" или "для всех", и квантор существования, которому соответствует символ $\exists$.

Квантор $\exists$ применяется, когда нужно сказать, что существует хотя бы одно некоторое значение переменной, для которой истинно данное утверждение.

Если применяется квантор общности, то говорят, что утверждение истинно для всех *X* в некотором множестве. Иными словами, если подставить вместо *X* любое значение, то утверждение будет истинно.

Если в утверждение входят несколько кванторов, то приходится учитывать их взаимное расположение.

Пользуясь кванторами, можно представить предложение:

Все специалисты по ЭВМ является программистами.

следующим образом:

$\forall$ *(X) специалист по ЭВМ (X)* $\rightarrow$ *программист(X).*

А предложение

Некоторые специалисты по ЭВМ является оптимистами.

так:

$\exists$ *(X) специалист по ЭВМ (X)* $\rightarrow$ *оптимист(X).*

Порядок, в соответствии с которым вводятся квантифицируемые переменные, может влиять на смысл утверждения. Например, выражение

$\forall$ *(X)* $\exists$ *(Y) служащий(X)* $\rightarrow$ *руководитель(Y,X).*

может быть интерпретировано так:

У каждого служащего есть руководитель.

Если же изменить порядок следования кванторов, например

$\exists \forall$ *(X)* $\forall$ *(Y) служащие(X)* $\rightarrow$ *руководитель(Y,X).*

то изменится и утверждение

Есть такое лицо, которое руководит всеми.

В исчислении предикатов существует много различных правил вывода. Они могут применяться либо для установления истинности утверждения в целом, либо для порождения заключения. Вот некоторые правила вывода:

1. *Modus ponendo ponens*: $A \rightarrow B$, $A \vdash B$ $\vdash$ означает *верно, что* или *имеет место*.

2. *Modus tollendo tollens*: $A \rightarrow B$, $\sim A \vdash \sim B$ (например, ЕСЛИ моя программа правильна, ТО она будет работать. Моя программа НЕ правильна, СЛЕДОВАТЕЛЬНО, она НЕ будет работать).

3. Двойное отрицание: $A \vdash \sim (\sim B)$ (например, моя программа работала СЛЕДОВАТЕЛЬНО, моя программа НЕ НЕ правильна).

4. Введения конъюнкции: $A, B \vdash \{A\&B\}$ (например, моя программа работала, она правильна, СЛЕДОВАТЕЛЬНО, моя программа работала И она правильна).

5. *Reductio ad absurdum*: $A \rightarrow B, A \rightarrow \sim B \vdash \sim A$ (например, ЕСЛИ моя программа правильна, ТО она будет работать, ЕСЛИ моя программа правильна ТО она не будет работать, СЛЕДОВАТЕЛЬНО, моя программа не правильна).

6. Специализации: $\forall X\ W(X), A \vdash W(X)$ (например, все предметы, являющимися компьютерами, не надежны, Notebook – компьютер, СЛЕДОВАТЕЛЬНО, Notebook ненадежен).

Раскрывая скобки и приводя подобные члены, всегда можно привести формулу исчисления высказываний к одной из двух форм:

1. Конъюнктивная нормальная форма. Формула записана в виде конъюнкции дизъюнктов (C_1 И C_2 И... C_N). Каждый дизъюнкт является дизъюнкцией атомарных предложений или отрицаний таких предложений (P_1 ИЛИ НЕ P_2 ИЛИ ... P_M). Это помогает в рассуждениях, так как истинность всей формулы означает истинность каждого дизъюнкта в отдельности, что облегчает рассмотрение формулы.

2. Дизъюнктивная нормальная форма. Формула записана в виде дизъюнкции выражений, каждое из которых является конъюнкцией атомарных предложений или их отрицаний.

Пусть, например, нужно написать, что два члена комитета, состоящего из трех человек (Андерсон, Бейнс и Кларк), голосовали за некоторое предложение, а один голосовал против. Мы можем символически записать *Андерсон голосовал за* в виде *A*, *Бейнс голосовал за* в виде *B*, и *Кларк голосовал против* в виде НЕ *C*. Выражая это в виде составного предложения в дизъюнктивной нормальной форме, можно написать

(*A* И *B* И НЕ *C*) ИЛИ (НЕ *A* И *B* И *C*) ИЛИ (*A* И *C* И НЕ *B*).

Каждое из выражений в скобках истинно лишь при одной конкретной комбинации значений *A, B* и *C*. Для любых значений *A, B* и *C*, отличных от этих комбинаций, все выражения в скобках будут ложными, так что и формула в целом будет ложной. Таким образом, если эта формула верна для голосования в комитете, то утверждения *A, B, C* не могут быть одновременно истинны, не могут быть одновременно ложны и более чем одно из них не может быть ложным.

4.2.2.Доказательство с введением допущения.

Для доказательства импликации вида $A \rightarrow B$ допускается, что левая часть *A* истинна, т.е. *A* принимается в качестве дополнительной посылки, и делаются попытки доказать правую часть, *B*. Сам метод опирается на две важные теоремы о доказательствах.

Теорема 4.1. $A \vdash B$ тогда и только тогда, когда $\vdash A \rightarrow B$.

Эта теорема утверждает, что доказуемость заключения *B* из допущения *A* эквивалентна доказуемости импликации $A \rightarrow B$ без каких-либо дополнительных допущений.

Теорема 4.2. $A_1, A_2, ..., A_n \vdash B$ тогда и только тогда, когда

$$\vdash (A_1 \wedge A_2 \wedge .. . \wedge A_n) \rightarrow B.$$

Эта теорема получается из предыдущей и того факта, что все посылки $A_1, ..., A_n$ истинны тогда и только тогда, когда истинна их конъюнкция (основное свойство связки И).

Наконец, полезная эквивалентность $(X \rightarrow (Y \rightarrow Z)) \leftrightarrow (X \wedge Y \rightarrow Z)$.

Она легко доказывается с помощью соотношений булевой алгебры, так как левая и правая части сводятся к $\sim X \vee \sim Y \vee Z$.

Рассмотрим пример доказательства с введением допущения.

Если $A_1, A_2, ..., A_n, P \vdash Q$, то $\vdash (A_1 \wedge A_2 \wedge \ldots \wedge A_n P) \rightarrow Q$ в силу теоремы

4.2, откуда $\vdash (A_1 \wedge A_2 \wedge \ldots \wedge A_n) \rightarrow (P \rightarrow Q)$ в силу эквивалентности, откуда $\vdash A_1, A_2, ..., A_n \vdash P \rightarrow Q$ в силу теоремы 2.

4.2.3. Доказательство приведением к противоречию.

При построении выводов не всегда целесообразно ждать появления искомого заключения, просто применяя правила вывода. Именно такое часто случается, когда делается допущение B для доказательства импликации $B \rightarrow C$. Применяем цепное правило и модус поненс к B и другим посылкам, чтобы в конце получить C. Однако можно пойти по неправильному пути, и тогда будет доказано много предложений, большинство из которых не имеют отношения к нашей цели. Этот метод носит название прямой волны и имеет тенденцию порождать лавину промежуточных результатов, если его запрограммировать для компьютера и не ограничить глубину.

Другая возможность использовать одну из приведенных выше эквивалентностей и попытаться, например, доказать $\sim C \rightarrow \sim B$ вместо $B \rightarrow C$. Тогда допустим $\sim C$ и попробуем доказать $\sim B$. Т.е. допускается, что заключение C (правая часть исходной импликации) неверно, и делается попытка опровергнуть посылку B. Это позволяет двигаться как бы назад от конца к началу, применяя правила так, что старое заключение играет роль посылки. Такая организация поиска называется *поиском от цели.*

Можно использовать также комбинацию этих методов, называемую приведением к противоречию. В этом случае для доказательства $B \rightarrow C$ допускаем одновременно B и $\sim C$, т.е. предполагаем, что заключение ложно:

$$\sim (B \rightarrow C) = \sim (\sim B \vee C) = B \wedge \sim C.$$

Теперь можно двигаться и вперед от B, и назад от $\sim C$. Если C выводимо из B. то, допустив B, доказали бы C. Поэтому допустив C, получим противоречие. Если же выведем $\sim B$ из $\sim C$, то тем самым получим противоречие с B. В

общем случае можно действовать с обоих концов, выводя некоторое предложение P, двигаясь вперед, и его отрицание $\sim P$, двигаясь назад. В случае удачи это доказывает, что посылки несовместимы, или противоречивы. Отсюда выводим, что дополнительная посылка $B \wedge \sim C$ должна быть ложна, а значит противоположное ей утверждение $B \wedge C$ истинно.

Следует отметить, что если предложение $B \rightarrow C$ ложно, то никогда не получим противоречия, сколько бы ни рассуждали. При автоматизации поиска доказательства здесь снова возникает проблема, так как компьютер не может сказать, когда нужно остановиться.

4.2.4. Доказательство методом резолюции.

Применяется всего одно правило вывода, что позволяет не запоминать многочисленных правил вывода и тавтологии. Это правило резолюции, которое приведено в табл.4.1 вместе с уже известными правилами.

Таблица 4.1 – Сравнение методов вывода

Резолюция	Цепное правило	Модус поненс
Из $X \vee A$	Из $\sim X \rightarrow A$	A
и $Y \vee \sim A$	и $A \rightarrow Y$	$A \rightarrow Y$
получаем $X \vee Y$	получаем $\sim X \rightarrow Y$	Y

Оно позволяет соединить две формулы, удалив из одной атом A, из другой $\sim A$. Из приведенной выше таблицы видно, что правило резолюции можно рассматривать как аналог цепного правила в применении к формулам, находящимся в конъюнктивной нормальной форме В данном случае цепное правило записано в форме, эквивалентной приведенной ранее, но более удобной для сопоставления с правилом резолюции. Правило *модус поненс* можно считать частным случаем правила резолюции для случая ложного X.

Правило резолюции применяем следующим образом.

Используем доказательство от противного и допускаем отрицание заключения.

Приводим все посылки и отрицание заключения, принятого в качестве дополнительной посылки, к конъюнктивной нормальной форме.

а) Устраняем символы ←→и → с помощью эквивалентностей

$$(B) = (A \rightarrow B) \vee C) \wedge (B \rightarrow A), \quad A \rightarrow B = \sim A \vee B.$$

б) Продвигаем отрицания внутрь с помощью законов де Моргана.

$$\sim (A \wedge B) = (\sim A) \vee (\sim B), \quad \sim (A \vee B) = (\sim A) \wedge (\sim B).$$

в) Применяем дистрибутивность

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C).$$

Этот метод приведения формул к виду, удобному для применения метода резолюции, обладает рядом серьезных недостатков, так как применение дистрибутивности, часто приводит к экспоненциальному разбуханию испытуемой формулы и разрушает ее структуру. В пропозициональном случае лучше привести к конъюнктивной нормальной форме саму исходную формулу, что сразу решает вопрос о ее тавтологичности.

Иногда целесообразнее упрощать формулу к виду, постепенно уменьшая ее глубину путем введения новых атомов, обозначающих ее части.

Теперь каждая посылка превратилась в конъюнкцию дизъюнктов, быть может, одночленную. Выписываем каждый дизъюнкт с новой строки; все дизъюнкты истинны, так как конъюнкция истинна по предположению.

Каждый дизъюнкт это дизъюнкция (возможно, одночленная), состоящая из предложений и отрицаний предложений. Именно к ним применим метод резо-

люций. Берем любые два дизъюнкта, содержащие один и тот же атом, но с противоположными знаками, например

$$X \vee Y \vee Z \vee \sim P, \quad X \vee P \vee W.$$

Здесь P – как раз тот атом, о котором шла речь. Применяем правило резолюции с атомом P в роли A из этого правила, т.е. отрезаем P от двух данных дизъюнктов:

$$(X \vee Y \vee Z) \vee (X \vee W) = X \vee Y \vee Z \vee W.$$

Это правило легко применять, так как оно сводится к простому удалению членов дизъюнктов.

Из $X \vee A$, $\sim A \vee Y$ выводим $X \vee Y$.

Из A, $\sim A \vee Y$ выводим Y.

Продолжаем этот процесс, пока не получится P и $\sim P$ для некоторого атома P. Применяя резолюцию и к ним, получим пустой дизъюнкт, выражающий противоречие, что завершает доказательство от противного.

Из P, $\sim P$ выводим ложь.

Пустой дизъюнкт обычно записывается в виде квадрата □.

В качестве примера рассмотрим доказательство соотношения:

$$P \vee Q,$$

$$P \rightarrow R,$$

$$Q \rightarrow S \vdash R \vee S.$$

1. Приводим посылки к нормальной форме и выписываем их на отдельных строках.

$$P \vee Q, \tag{4.1}$$

$$\sim P \vee R, \tag{4.2}$$

$$\sim Q \vee S. \tag{4.3}$$

2. Записываем отрицание заключения и приводим его к нормальной форме.

$$\sim (R \vee S) = \sim R \wedge \sim S,$$

$$\sim R, \tag{4.4}$$

$$\sim S. \tag{4.5}$$

3. Выводим пустой дизъюнкт с помощью резолюции. Из (4.4) и (4.2) получаем

$$\sim P. \tag{4.6}$$

Из (4.6) и (4.1) будет

$$Q. \tag{4.7}$$

Из (4.5) и (4.3) имеем

$$\sim Q. \tag{4.8}$$

Из (4.7) и (4.8) получим

$$\square. \tag{4.9}$$

Знак □ означает, что доказываемое предположение верно.

По сравнению с классической логикой метод резолюции имеет ряд преимуществ.

Не приходится применять эквивалентности для того, чтобы переставлять члены дизъюнкции $P \vee Q$ для получения $Q \vee P$ и т.д. Это происходит отчасти потому, что все приводится к нормальной форме с самого начала, а отчасти потому, что для правила резолюции неважно положение отрезаемого атома в

дизъюнкте.

Нужно помнить всего одно правило.

Применение этого правила легко автоматизировать, т.е. запрограммировать для компьютера. Однако в случае длинного доказательства легко зациклиться, а однородные обозначения делают все дизъюнкты похожими друг на друга, так что трудно удерживать в памяти их смысл и выбирать нужный дизъюнкт.

4.3. Представление знаний предметной области

На сегодняшний день существует три способа представления знаний. Это семантически сети, фреймы и продукционные правила.

4.3.1. Семантические сети

Понятие семантической сети основано на простой идее о том, что память формируется через ассоциации между понятиями. Понятие ассоциативная память появилось еще во времена Аристотеля и вошло в информатику в связи с работами по использованию простых ассоциаций для представления значений слов в базе данных.

С тех пор этот формализм был всесторонне развит для представления многих классов данных, используемых в различных предметных областях. К таким областям относятся пространственные связи в простых физических системах, операции по управлению механизмами, причинные и функциональные связи в приборах и взаимосвязи между симптомами в медицине [6-12].

Базовым функциональным элементом семантической сети служит структура из двух компонентов – <узлов> и связывающих их <дуг>. Каждый узел представляет некоторое понятие, а дуга отношение между парами понятий.

Можно считать, что каждая из таких пар отношений представляет про-

стой факт. Узлы помечаются именем соответствующего отношения. Рис. 4.3, например, представляет факт <Антон работает в производственном отделе>. Дуга имеет направленность, благодаря чему между понятиями, в рамках определенного факта, выражается отношение <субъект/объект>. Более того, любой из узлов может быть соединен с любым числом других узлов; в результате этого обеспечивается формирование сети фактов.

С позиций логики базовую структуру семантической сети можно рассматривать в качестве эквивалента предиката с двумя аргументами (бинарный предикат); эти два аргумента представляются двумя узлами, а собственно предикат – направленной дугой, связывающей эти узлы. Пользуясь подобными отношениями, можно представлять сложные совокупности фактов.

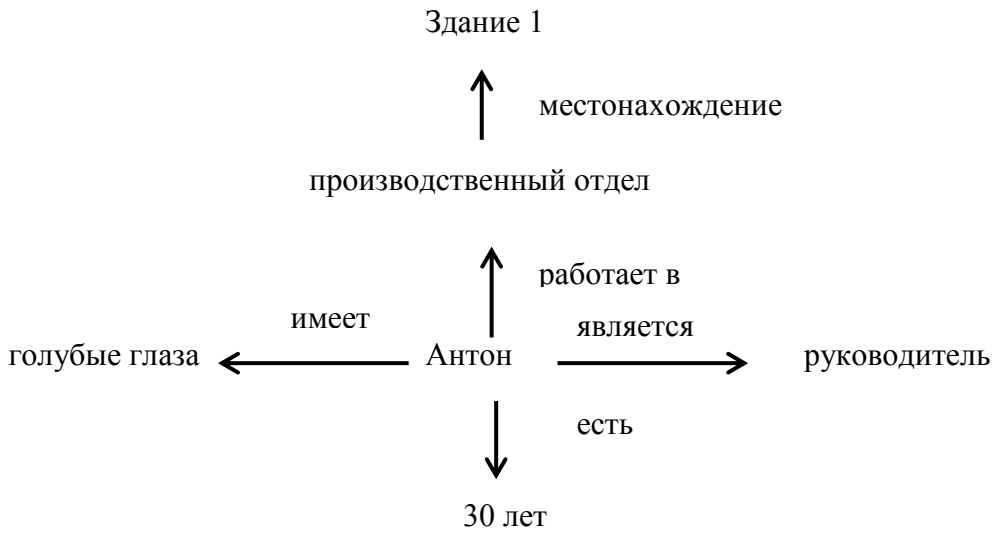


Рис. 4.3. Факты о человеке Антон

Популярность семантических сетей обязана связи <является>, в которой заложены большие возможности для построения иерархий понятий. Пример такой иерархии узел <служащий> на рис. 4.4. Узлы <Ан-1> и <Ан-2> позволяют описать двух разных людей с одинаковым именем.

Иерархия, построенная на основе наследования, обеспечивает эффективный способ упрощения представления знании и сокращения объема информации, которую требуется запоминать для каждого конкретного узла. Это дает

возможность в значительной мере ускорить процесс обработки знаний (относящаяся к узлу запоминаемая информация может быть ограничена только часто используемой; при обращении к остальной информации применяется принцип наследования), а также извлекать информацию с помощью запросов общего характера (некоторая информация об индивидууме <Антон> как руководителе может быть извлечена просто из знания его служебного положения в компании; при этом нет необходимости знать его имя).

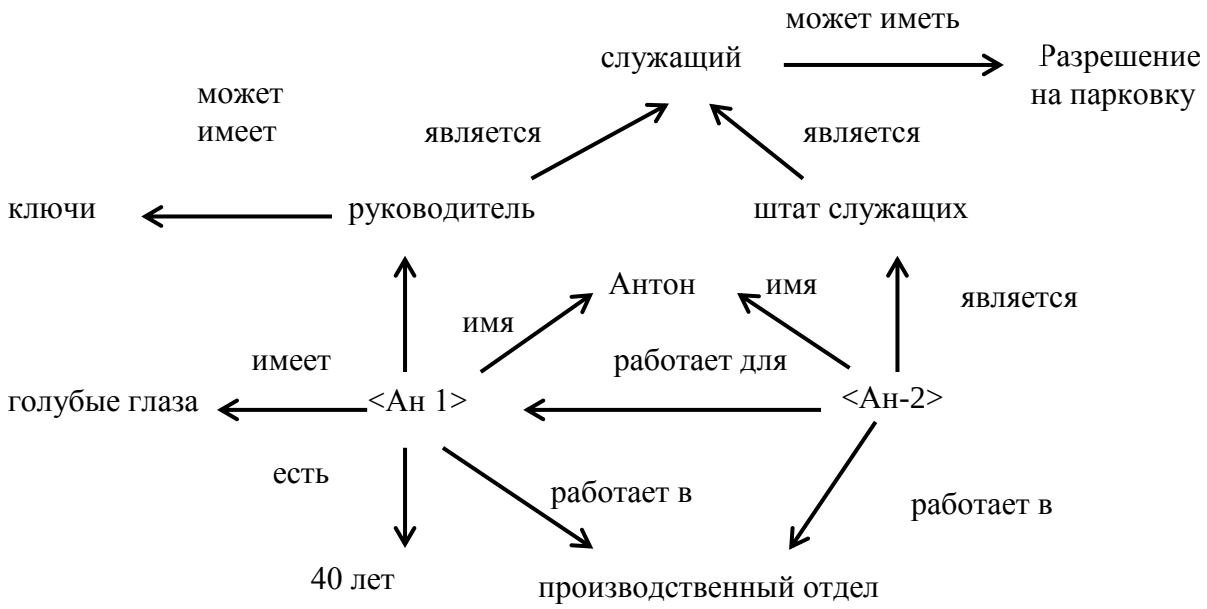


Рис. 4.4. Факты, относящиеся к служащим <Ан 1> и<Ан 2>

Приведенные примеры представления знаний в виде семантических сетей ограничены отношениями между существительными или фразами, составленными из существительных. Нужно только разработать представление относительно глаголов в дополнение к существительным.

Были разработаны непротиворечивые полные наборы вербальных отношении. Они получили название падежных отношений в соответствии с теорией падежной грамматики, разработанной Ч. Филлмором. В этой теории предпринимается попытка представить поверхностную структуру предложений в виде

небольших замкнутых наборов <падежных> отношении между существительными (или фразами из существительных) и глаголами в рамках глубинных структур предложений. Примерный набор подобных отношении может в типичном случае включать: агента – исполнителя (инициатора) действия, выражаемого глаголом; объекта – имя, на которое распространяется действие или состояние, выражаемое глаголом; местоположение – место действия или состояние; датив (dative) – лицо, к которому имеет отношение действие или состояние.

4.3.2. Фреймы.

Принцип организации свойств некоторого объекта или события для формирования прообраза реализуется с помощью нотации вида <фрейм>. Для иллюстрации этого опишем сведения о служащем компании, как это сделано в предыдущем разделе. Достоинство системы, использующей фреймы, заключается в том, что те элементы, которые традиционно присутствуют в описании объекта или события, группируются и благодаря этому могут извлекаться и обрабатываться как единое целое. Первый пример касается понятия <руководитель> (рис. 4.5) и иллюстрирует некоторые особенности фреймов.

имя: РУКОВОДИТЕЛЬ
специальность: СЛУЖАЩИЙ
имя: агрегат (фамилия, имя, отчество)
возраст; агрегат (годы)
адрес: АДРЕС
отдел: диапазон (производство, администрация)
заработная плата: ЗАРПЛАТА

Рис. 4.5. Фрейм для понятия <РУКОВОДИТЕЛЬ>

Во-первых, фрейм имеет имя для идентификации описываемого им понятия. Во-вторых, его описание составляется из ряда описаний, приведенных на рисунке слева, которые получили название <слоты>. С помощью слотов иден-

тифицируются основные структурные элементы понятий. За слотами следуют шпации (промежутки), в которые помещают некоторые объекты, представляющие текущие значения слотов.

На рис. 4.5 дан фрейм с заполненными слотами. При этом часть из них заполнена некими объектами, а не простыми именами. В данном примере фигурируют три различных типа таких заполнителей слотов. Заполнитель слота может быть или константой, или именем другого фрейма.

Простейшими из них являются те, что представлены прописными буквами (например, АДРЕС, ЗАРПЛАТА). Это имена других фреймов данной системы, на которые делается ссылка. Кроме того, существуют обозначения <агрегат> и <интервал>. В процессе обработки систем фреймов иногда необходимо наложить ограничения на тип объекта, который может быть использован для заполнения некоторого слота. Обозначение <агрегат> указывает на то, что должны быть заданы определенные объекты, а обозначение <диапазон> – на то, что должен быть выбран один из объектов.

Существует ряд языков, специально разработанных для облегчения конструирования на основе фреймов различных процедур обработки знаний, например, Лисп.

4.3.3. Правила продукций.

Самым распространенным форматом для представления знаний, наиболее соответствующим их процедурному характеру, является правило продукции, которое по своей сути – просто программа из одного оператора вида

<ЕСЛИ условие, ТО действие>.

Люди имеют запас знаний о мире, в котором они живут. Знания обычно представляются в виде фактов, характерных для окружающего мира (т. е. классов объектов и взаимосвязей между ними), процедур и правил манипулирования фактами, а также в виде информации о том, когда и как следует применять

правила и процедуры.

Объекты группируют по классам. Петр, Джон, Фред и Анна могут быть объектами. Их можно отнести к классу “личность”. В дополнение к этому Петр, Джон и Фред могут быть классифицированы как “мужчины”, а Анна – как “женщина”. Явное достоинство любой классификации заключается в том, что частично решается проблема переполнения памяти, так как достаточно помнить только характеристики класса, а не каждого объекта. Можно также определить отношения между классами (или отдельными объектами). Подобным образом можно определить отношение “руководит(*A*, *B*)”, означающее, что *B* находится в подчинении у *A*. В качестве примеров такой зависимости могут служить выражения:

руководит (Петр, Джон),
руководит (Джон, Анна),
руководит (Анна, Фред),

которые заключают в себе структуру “подотчетность” между объектами Петр – Джон – Анна – Фред. Приведенный пример иллюстрирует отношения между схожими объектами. Между различающимися объектами также могут быть установлены отношения (например, “владеет (Петр, автомобиль)”). Знания об объектах и их взаимоотношениях позволяют классифицировать эти объекты и соотносить между собой.

Еще один тип знаний – правила. Они дают возможность определить, как вывести новые отличительные особенности класса или отношения для объектов, прежде не подразделенных на классы. Например, если определить отношение “отчитывается (*B*, *A*)” для того, чтобы отметить, что *B* подотчетен *A* (возможно, через других руководителей), то можно установить правила:

“отчитывается (*C*, *A*)” есть ИСТИНА,
ЕСЛИ или “руководит (*A*, *C*)” есть ИСТИНА
ИЛИ “руководит (*A*,*B*)” И “руководит (*B*, C)" есть ИСТИНА.

Это ограниченное правило применимо только для первого или второго уровня подотчетности, но в пределах ограничения оно дает нам возможность породить новый пример отношения “отчитывается”, который ранее не был известен. Например, правило позволяет сделать заключение о том, что “отчитывается (Анна, Петр)” и “отчитывается (Фред, Джон)” суть ИСТИНА. Вследствие того, что данное правило определено как двухуровневое, оно не может быть применено для получения вывода “отчитывается (Фред, Петр)” есть ИСТИНА. Для этого нам потребуется более мощное правило, включающее рекурсию:

“отчитывается (*C*, *A*)” есть ИСТИНА,
ЕСЛИ или “руководит (*A*, *C*)” есть ИСТИНА
ИЛИ “руководит (*A*, *B*)” есть ИСТИНА
И “отчитывается (*C*, *B*)” есть ИСТИНА.

Первая часть приведенного рекурсивного правила относится к прямой подотчетности, а вторая – к косвенной.

Если задать вопрос “отчитывается (Джон, Петр)?”, то ответом будет ИСТИНА, поскольку истинна первая часть правила “ЕСЛИ”. Вопрос “отчитывается (Фред, Петр)?” потребует более сложной обработки. Табл. 4.2 показывает процесс подобной обработки с использованием рассмотренного правила.

Таблица 4.2 – Пример отношения “отчитывается”

№	Вопрос	Первое ИЛИ	Второе ИЛИ	Новый вопрос
1	Отчитывается (Фред, Петр)?	руководит(Петр, Фред)? ложь	руководит(Петр, B)? истина B= Джон	отчитывается (Фред, Джон)?
2	Отчитывается (Фред, Джон)?	руководит(Джон, Фред)? ложь	руководит (Джон, B)? истина B= Анна	отчитывается (Фред, Анна)?
3	Отчитывается (Фред, Анна)?	руководит (Анна, Фред)? истина		

Поэтому отчитывается (Фред, Анна) есть истина. Так как отчитывается (Фред, Джон) есть истина. Поэтому отчитывается (Фред, Петр) есть истина,что и служит ответом на вопрос.

Отметим рекурсивность обработки отношения “отчитывается”.

Следующая необходимая компонента процесса обработки знаний – управляющая структура, определяет способ применения разнообразных правил тем самым определяя стратегию решения. Данные частных случаев можно хранить в рабочей памяти.

Механизм вывода осуществляет цикл “распознавание-действие”, а управление может осуществляться на основе данных или от цели. Работу этого механизма вывода представляют последовательностью из трех шагов.

1. Интерпретатор сопоставляет образец правила с элементами данных в базе знаний.

2. Если можно вызвать более одного правила, то интерпретатор использует механизм разрешения конфликта для выбора правила.

3. Интерпретатор применяет выбранное правило, чтобы найти ответ.

Этот трехшаговый процесс интерпретации является циклическим и называется циклом “распознавание-действие”.

В системе, базирующейся на правилах, результат является действием одного из продукционных правил, которые определяются входными данными.

Экспертная система также содержит интерпретатор в механизме вывода, который выбирает и активизирует различные модули системы.

В заключение перечислим *преимущества продукционных систем*:

- отделение знаний от программы поиска;
- модульность правил (правила не могут вызывать другие правила);
- возможность эвристического управления поиском;
- возможность трассировки “цепочки рассуждений”;
- независимость от выбора языка программирования;
- продукционные правила являются правдоподобной моделью решения задачи человеком.

4.4. Методы поиска решений в пространстве состояний

Поиск в пространстве состояний можно вести в двух направлениях: от исходных данных задачи к цели и в обратном направлении от цели к исходным данным.

При поиске на основе данных, который иногда называют *прямой цепочкой*, начинают процесс решения задачи, анализируя ее условие, а затем применяют допустимые ходы или правила изменения состояния. В процессе поиска правила применяются к известным фактам для получения новых фактов, которые, в свою очередь, используются для генерации новых фактов. Этот процесс продолжается до тех пор, пока, если повезет, не будет достигнута цель.

Возможен альтернативный подход. Рассмотрим цель, которую хотим достичь. Проанализируем правила или допустимые ходы, ведущие к цели, и определим условия их применения. Эти условия становятся новыми целями, или подцелями, поиска. Поиск продолжается в обратном направлении от достигнутых целей до тех пор, пока (если повезет) не достигнем исходных данных задачи. Этот подход называется *поиском от цели* или *обратной цепочкой*.

В обоих случаях обрабатывается один и тот же граф, однако порядок и число состояний в процессе поиска могут различаться. Какую стратегию поиска предпочесть зависит от самой задачи. При этом следует учитывать сложность правил, природу и доступность данных задачи. Выбор зависит также от структуры решаемой задачи.

Процесс поиска от цели рекомендован в следующих случаях:

1. Цель поиска явно присутствует в постановке задачи или может быть легко сформулирована.

2. Имеется большое число правил, которые на основе полученных данных позволяют генерировать возрастающее число заключений или целей. Своевременный отбор целей позволяет отсеять множество возможных ветвей, что делает процесс поиска более эффективным.

3. Исходные данные не приводятся в задаче, но подразумевается, что должны быть известны исследователю. В этом случае поиск от цели может служить руководством для правильной постановки задачи.

Поиск на основе данных применим к решению задачи в случаях:

1. Все или большинство исходных данных заданы в постановке задачи.

2. Существует большое число потенциальных целей, но имеется всего лишь несколько способов применения фактов и предоставления информации о конкретном примере задачи.

3. Сформировать цель или гипотезы очень сложно.

Поиск в двух направлениях одновременно называют *двунаправленным поиском*.

4.4.1. Реализация поиска на графе.

Поиск с возвратом (bactracing) – это метод систематической проверки различных путей в пространстве состояний. Алгоритм поиска с возвратом запускается из начального состояния и следует по некоторому пути до тех пор, пока не достигнет цели или не упрется в тупик. Если поиск привел в тупиковую вершину, но алгоритм *возвращается* в ближайшую из пройденных вершин и исследует все ее вершины-братья, а затем спускается по одной из ветвей, ведущих от вершины-брата. Если цель достигнута, то поиск завершается, и в качестве решения задачи возвращается путь к цели. Алгоритм работает до тех пор, пока не достигнет цели или не исследует все пространство состояний.

Поиск в глубину. При поиске в глубину после исследования состояния сначала необходимо оценить всех его потомков, а затем исследовать любую из вершин-братьев. Алгоритм поиска с возвратом осуществляет поиск в глубину.

Поиск в ширину исследует пространство состояний по уровням, один за другим. И только если состояний на данном уровне больше нет, алгоритм переходит к следующему уровню.

Поиск в ширину гарантирует нахождение кратчайшего пути от начального состояния к цели. Поиск в глубину не гарантирует нахождения оптимального пути к состоянию, если оно встретилось впервые. Позже в процессе поиска могут быть найдены различные пути к любому состоянию. Если длина пути имеет значение в решении задачи, то необходимо сохранить именно тот путь, который оказался короче.

4.5. Алгоритмы эвристического поиска

При значительном числе состояний время поиска оптимального пути возрастает экспоненциально, и в этом случае могут помочь *алгоритмы эвристического поиска*, которые обладают высокой вероятностью правильного выбора решения. Рассмотрим некоторые из этих алгоритмов.

4.5.1. Алгоритм наискорейшего спуска по дереву решений.

Пример построения более узкого дерева рассмотрим на примере задачи о коммивояжере. Торговец должен побывать в каждом из 5 городов, обозначенных на карте (рис. 4.6) [6, 9].

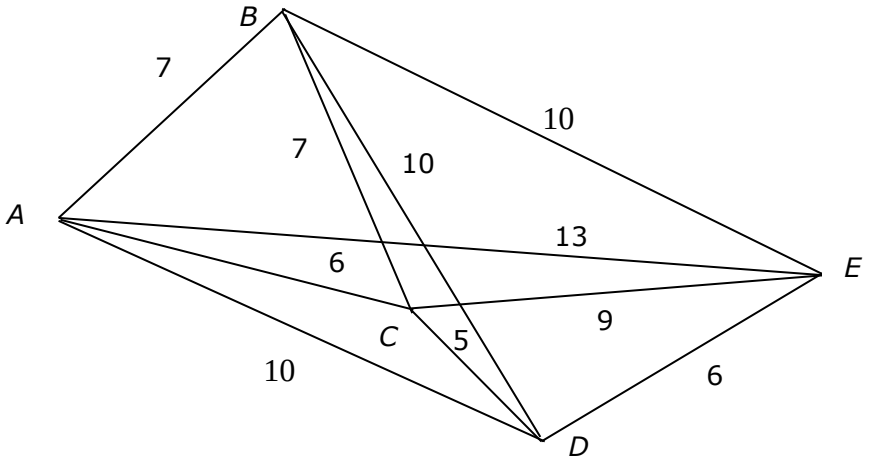


Рис. 4.6. Условная карта городов

Задача состоит в том, чтобы, начиная с города *A*, найти минимальный путь, проходящий через все остальные города только один раз и приводящий обратно в *A*. Идея метода проста – из каждого города идем в ближайший, где еще не были. Решение задачи показано на рис. 4.7.

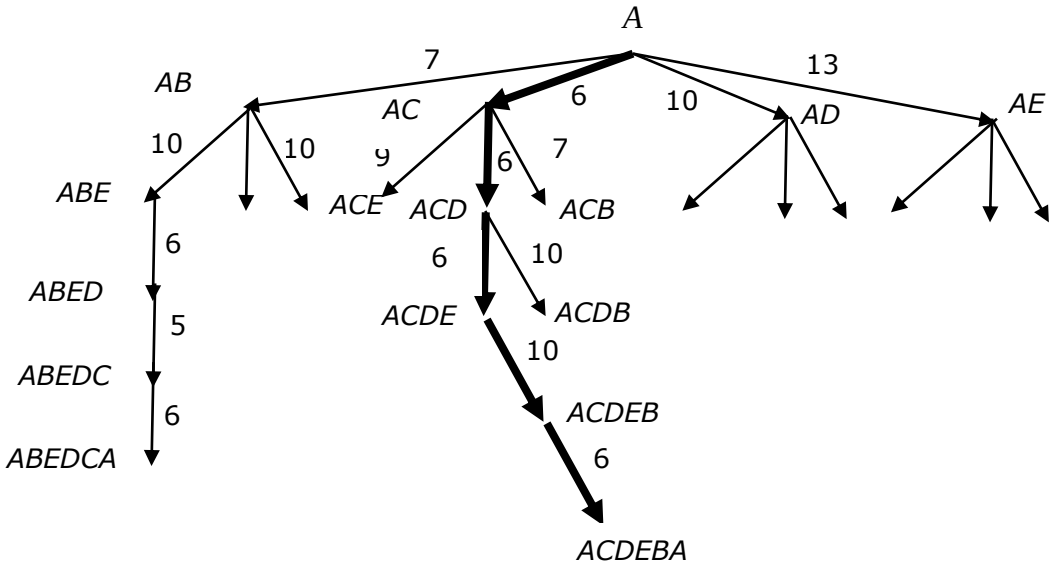


Рис. 4.7. Дерево решения задачи

Такой алгоритм поиска решения получил название *алгоритма наискорейшего спуска* (в некоторых случаях – наискорейшего подъема).

Однако этот алгоритм не всегда гарантирует лучшее решение. Например, путь *ABEDCA* имеет такую же длину, как и найденный алгоритмом, но этот путь получается в случае, когда на первом и втором шагах выбираются не самые близкие города.

4.5.2. Оценочные(штрафные) функции.

В игре “крестики-нолики” есть два игрока MAX (играющий фишками X) и MIN (фишками 0). Значение оценки для доступных им ходов могут быть получены путем подсчета всех линий, открытых для каждого из игроков, а затем вычитания одного числа из другого (рис. 4.8).

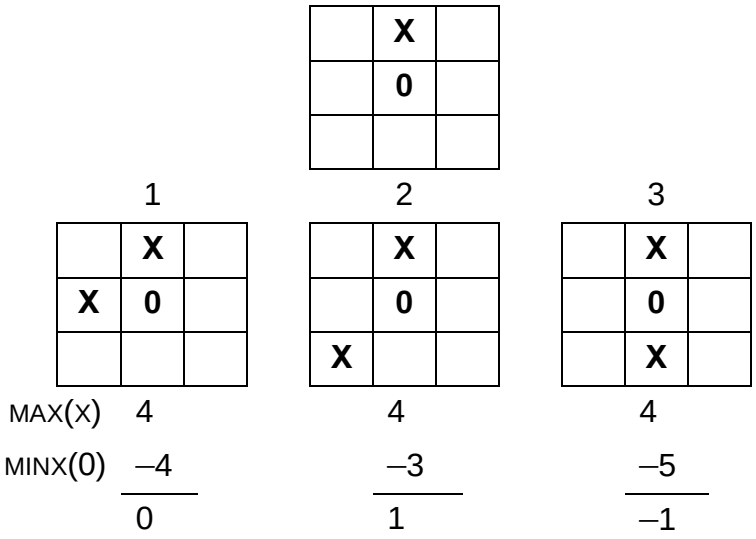


Рис. 4.8.Применение эвристической оценочной функции

Применительно к позиции MAX должен сделать выбор среди вариантов 1, 2 и 3. Они соответственно получают оценки 0, 1, –1. MAX выбирает вариант 2, так как он получил наивысшую оценку для этого раунда.

В игре в “крестики-нолики” приемлема конструкция единственной оценочной функции, которая надежно прокладывает путь к оптимальному решению и вычисляется с помощью алгоритма минимакса, оптимизирующего на каждом шаге оценочную функцию игрока, делающего свой ход.

4.5.3.Алгоритм минимакса.

Оскар Моргенштерн и Джон фон Нейман предложили *метод минимакса*, нашедший широкое применение в теории игр [6, 9]. Предположим, что противник использует оценочную функцию, совпадающую с нашей оценочной функцией. Выбор хода с нашей стороны определяется максимальным значением оценочной функции для текущей позиции. Противник стремится сделать ход, который минимизирует свою оценочную функцию. Поэтому этот метод и получил название минимакса.

На рис. 4.9 приведен пример анализа дерева ходов с помощью *метода минимакса* (выбранный путь решения отмечен жирной линией).

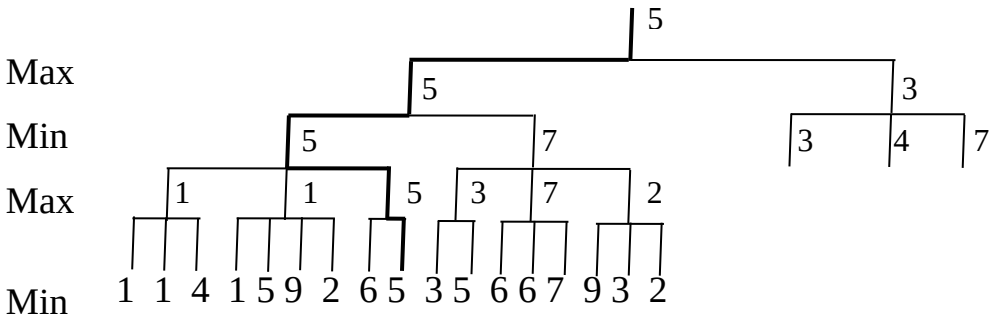


Рис. 4.9. Дерево ходов

4.5.4.Альфа-бета процедура.

Теоретически это эквивалентная минимаксу процедура, с помощью которой всегда получается такой же результат, но заметно быстрее, так как целые части дерева исключаются без проведения анализа. В основе этой процедуры лежит идея Дж. Маккарти об использовании двух переменных, обозначенных α и β [6, 8, 9]. Основная идея метода состоит в сравнении наилучших оценок, полученных для полностью изученных ветвей, с наилучшими предполагаемыми оценками для оставшихся. Можно показать, что при определенных условиях некоторые вычисления являются лишними. Рассмотрим идею отсечения на примере рис. 4.10.

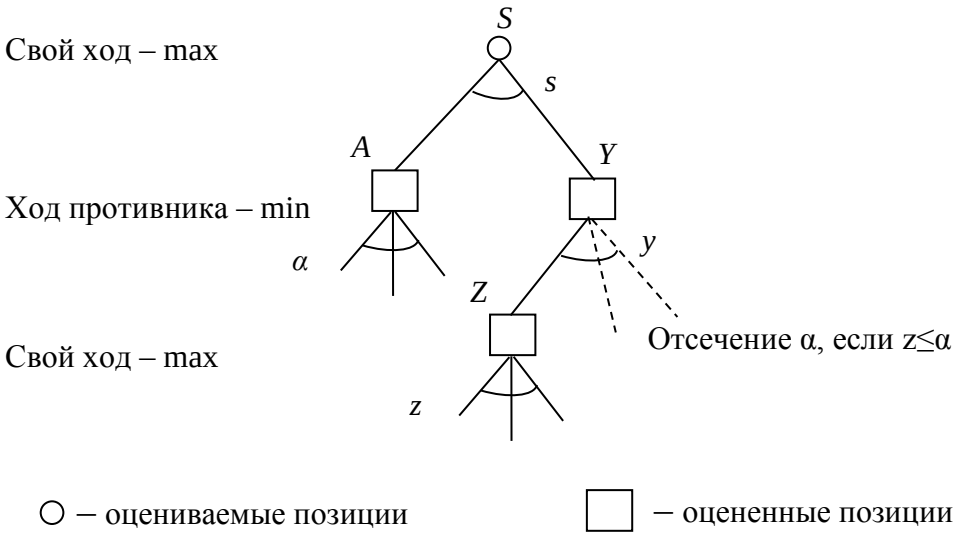


Рис. 4.10. α-отсечение

Предположим, позиция *A* полностью проанализирована и найдено значение ее оценки α. Допустим, что один ход из позиции *Y* приводит к позиции *Z*, оценка которой по *методу минимакса* равна *z*. Предположим, что $z \le \alpha$. После анализа узла *Z*, когда справедливо соотношение $y \le z \le \alpha \le s$, ветви дерева, выходящие из узла *Y*, могут быть отброшены (альфа-отсечение).

Если захотим опуститься до узла *Z*, лежащего на уровне произвольной глубины, принадлежащей той же стороне, что и уровень *S*, то необходимо учитывать минимальное значение оценки β, получаемой на ходах противника.

Отсечение типа β можно выполнить всякий раз, когда оценка позиции, возникающая после хода противника, превышает значение β. Алгоритм поиска строится так, что оценки своих ходов и ходов противника сравниваются при анализе дерева с величинами α и β соответственно. В начале вычислений этим величинам присваиваются значения $+\infty$ и $-\infty$, а затем, по мере продвижения к корню дерева, находится оценка начальной позиции и наилучший ход для одного из противников.

Правила вычисления α и β в процессе поиска рекомендуются следующие.

1) У MAX вершины значение α равно наибольшему в данный момент значению среди окончательных возвращенных значений для ее дочерних вершин;

2) У MIN вершины значение β равно наименьшему в данный момент значению среди окончательных возвращенных значений для ее дочерних вершин.

Правила прекращения поиска.

1. Можно не проводить поиска на поддереве, растущем из всякой MIN вершины, у которой значение β не превышает значения α всех ее родительских MAX вершин.

2. Можно не проводить поиска на поддереве, растущем из всякой MAX вершины, у которой значение α не меньше значения β всех ее родительских MIN вершин.

На рис. 4.11 показаны α-β отсечения для конкретного примера.

Таким образом, α-β алгоритм дает тот же результат, что и метод минимакса, но выполняется быстрее.

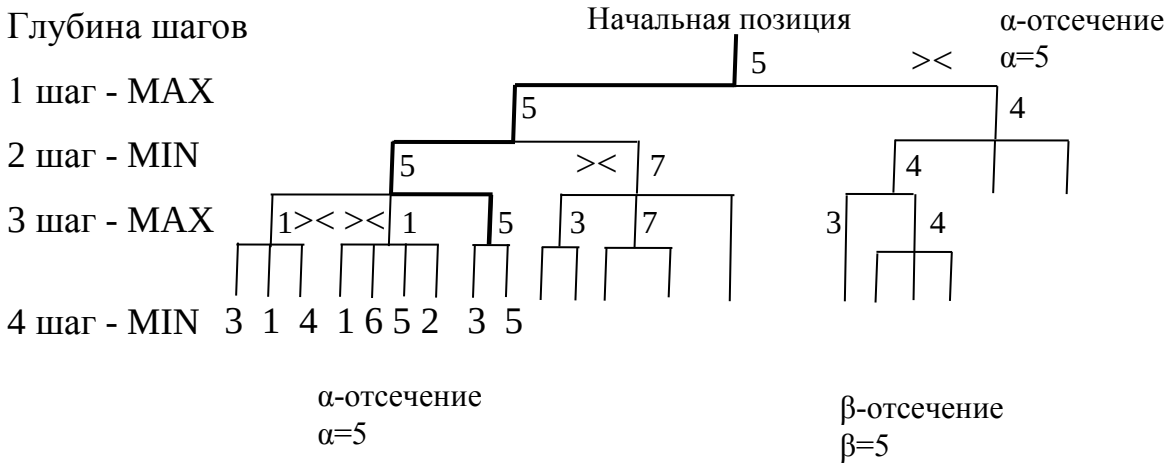


Рис. 4.11. β отсечение для конкретного примера

Использование *алгоритмов эвристического поиска* для поиска на графе И, ИЛИ выигрышной *стратегии* в более сложных задачах и играх (шашки, шахматы) не реально.

По некоторым оценкам игровое дерево игры в шашки содержит 10^{40} вершин, в шахматах 10^{120} вершин. Если при игре в шашки для одной вершины требуется 1/3 наносекунды, то всего игрового времени потребуется 10^{21} веков. В таких случаях вводятся искусственные условия остановки, основанные на таких факторах, как наибольшая допустимая глубина вершин в дереве поиска или ограничения на время и объем памяти.

Контрольные вопросы

1. Построить таблицы истинности для следующих выражений:

$\neg r \,\&\, \neg (\neg s \,\&\, \neg p) \to p$.

$\neg p \Leftrightarrow q \wedge \neg r$.

$((p \to q) \wedge (q \to r) \wedge p) \to (p \to r)$.

$(p \to \neg q) \wedge (s \wedge p \to s)$.

$\neg p \vee (\neg (p \wedge s) \to (\neg s \wedge r)$.

2. Найти значения следующих формул:

$(\neg p1 \to \neg(p2 \Leftrightarrow p3)) \wedge (\neg p1 \vee \neg p2 \vee \neg p3)$,

если $p1 = 1,\ p2 = 0,\ p3 = 1$;

$\neg (p1 \to (\neg p2 \to \neg(p3 \Leftrightarrow (\neg p1 \wedge (p2 \vee p3)))))$,

если $p1 = 0,\ p2 = 0,\ p3 = 1$;

$(p1 \wedge p2 \wedge p3) \to (p3 \vee (\neg p2 \to p1))$,

если $p1 = 1,\ p2 = 0,\ p3 = 0$;

$\neg (p1 \to \neg(p2 \vee (\neg p3 \wedge p2 \Leftrightarrow p3)))$,

если $p1 = 1,\ p2 = 1,\ p3 = 1$;

$(p1 \vee \neg p2) \to (p3 \to \neg(p2 \to \neg(p1 \vee \neg p2 \vee \neg p3)$,

если $p1 = 0,\ p2 = 0,\ p3 = 1$.

3. Записать символически высказывания, употребляя буквы для обозначения простых высказываний. Построить таблицы истинности для каждого высказывания:

3.1. Пётр ходит в кино только в том случае, когда показывают комедию.

3.2. Необходимое и достаточное условие для жизни растений состоит в наличии питательной почвы, чистого воздуха и солнечного света.

3.3. Студент не может заниматься, если он устал или голоден.

Когда вы устранили невозможное, все, что останется, даже невероятное, должно быть правдой.

Шерлок Холмс, “Знак четырех”

5. ВЫВОД В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ ИЛИ НЕПОЛНЫХ ЗНАНИЙ

5.1. Виды неопределенности

В предыдущих примерах все знания были определенными. Утверждениями были или ИСТИНА, или ЛОЖЬ. Однако в жизни имеется тенденция к “нечеткости” в представлении знаний. Тем не менее, на основании неточных данных часто можно делать вполне определенные умозаключения. Для этого приходится рассматривать комбинацию элементов знаний, а также значения их определенности и в результате выводить новые знания и давать оценку их определенности [6-12].

Человек делает необходимые умозаключения в подобных условиях ежедневно: ставит медицинские диагнозы и назначает лечение, руководствуясь симптомами; выясняет причины плохой работы двигателя по акустическому шуму; правильно понимает обрывки фраз естественного языка; узнает друзей по их голосам и т.п. Знаниям, которыми оперирует человек в указанных случаях, присуща неопределенность. Она может порождаться неполнотой описания ситуации, вероятностным характером наблюдаемых событий, неточностью представления данных, многозначностью слов естественного языка, использованием эвристических правил вывода и др. Наиболее важные виды неопределенности можно представить в виде дерева (рис. 5.1) [7, 8].

На первом уровне дерева изображены термины, представляющие качествен-

ную оценку характера неопределенности. Неопределенность может быть связана либо с неполнотой знаний, либо с их неоднозначностью.

Неполнота знаний возникает, когда собрана не вся информация, необходимая для построения выводов. Неоднозначность означает, что истинность техили иных высказываний не может быть установлена с абсолютной достоверностью. Она порождается либо физическими причинами (физическая неопределенность), либо лингвистическими (лингвистическая неопределенность).

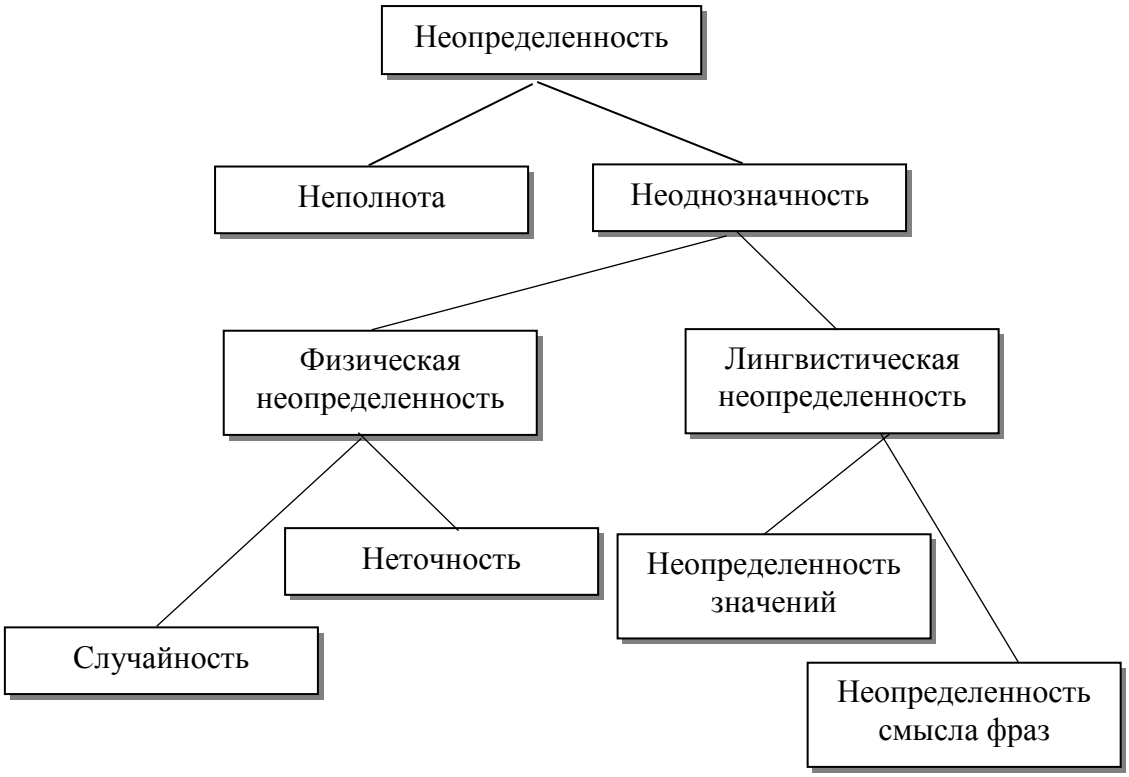


Рис. 5.1. Виды неопределенности

Физическая неопределенность может быть связана со случайностью событий, ситуаций, состояний объекта или неточностью представления данных. *Лингвистическая неопределенность* связана с использованием естественного языка для представления знаний, имеющих качественный характер, и возникает из-за множественности значений слов (полисемия) и смысла фраз. Например, “Двигатель часто перегревается” или “Ваня уже большой”. В этих примерах неоднозначность обусловлена нечеткостью понятий “часто” и “большой”. Или, например: “Он встретил ее на поляне с

цветами". Как он ее встретил: с цветами или без цветов? Такого рода неопределенности присутствуют в системах, на поведение которых в значительной степени влияют суждения человека. При анализе неопределенности смысла фраз выделяют синтаксическую, семантическую и прагматическую неопределенности [6-8].

Рассмотренная схема видов неопределенности условна. В реальных системах указанные виды неопределенности могут накладываться одна на другую. Например, физическая неопределенность может усложняться лингвистической.

Часто указанные выше виды неоднозначности не выделяют в отдельные группы, а рассматривают в рамках одного термина "ненадежные знания" [7, 8]. Основополагающим понятием, используемым при построении моделей вывода на таких знаниях, является понятие достоверности. Достоверность выводов на основе ненадежных знаний может быть определена с помощью различных подходов. Наиболее часто используются: вероятностная байесовская логика, коэффициенты уверенности, нечеткая логика, теория Демпстера – Шефера [1, 10].

Неполнота знаний приводит к необходимости осуществления немонотонных выводов. Для формальной обработки знаний, характеризуемых неполнотой, используются логика умолчания Рейтера, немонотонная логика Мак-Дермотта и Доила, системы поддержания значений истинности.

5.2. Байесовский метод

При байесовском подходе степень достоверности каждого из фактов базы знаний оценивается вероятностью, которая принимает значения в диапазоне от 0 до 1. Вероятности исходных фактов определяют либо методом статистических испытаний, либо опросом экспертов [10].

Рассуждения в условиях неопределенности имеют место и в системах наблюдения при наличии одновременно нескольких конкурирующих гипотез и их постоянной переоценке по мере поступления новых данных. В конце выделяется одна гипотеза, которая позволяет сделать соответствующий вывод. Так

работает система PROSPECTOR, применяемая при поиске рудных месторождений.

Системы распознавания речи также используют этот метод. Распознавание речи – сложная задача, поэтому в ней должны присутствовать конкурирующие гипотезы, например, о том, какое конкретное слово употребляется в предложении. На основе имеющихся гипотез с помощью различных источников неполной информации делается опосредованный вывод.

Если существует только одна применимая эвристика, то проблем нет. Но что делать, если два эвристических метода ориентируют программу в двух разных направлениях? А когда две эвристики указывают одно и то же направление поиска, должно ли это вызывать большее доверие, чем если бы была только одна из них?

Существуют четыре важные проблемы, которые необходимо обсудить применительно к понятию неопределенности в автоматических системах вывода:

1. Как количественно выразить степень определенности при установлении истинности (или ложности) некоторой части данных?

2. Как выразить степень поддержки заключения конкретной посылкой?

3. Как использовать совместно две (или более) посылки, независимо влияющие на заключение?

4. Как быть в ситуации, когда нужно обсудить цепочку вывода для подтверждения заключения в условиях неопределенности?

Во всех рассуждениях в условиях неопределенности используются четыре основных правила [1, 10].

Условная вероятность $p(A|B)$ события A при данном B – это вероятность того, что событие A наступит при условии, что наступило событие B. Например, вероятность того, что пациент действительно страдает заболеванием A, если у него обнаружен только симптом B,

$$p(A|B) = p\ (A \text{ и } B) / p(B) \text{ или } p(B|A) = p(A \text{ и } B) / p(A),$$

где $p(A$ и $B)$ – вероятность одновременного наступления событий A и B.

Это – основная формула условной вероятности.

Если для каждой формулы вычислить величину $p(A$ и $B)$ и приравнять результаты, то получится:

$$p(A)\ p(B|A) = p(B)\ p(A|B).$$

Эта формула известна как *правило Байеса*.

Следующее правило ИЛИ

$$p(A \text{ или } B) = p(A) + p(B) - p(A \text{ и } B).$$

Еще одно правило называется *правилом композиции*. В нем утверждается, что вероятность события A есть среднее взвешенное двух других вероятностей

$$p(A) = p(A|B)\ p(B) + p(A|\text{не } B)\ p(\text{не } B).$$

Рассмотрим первую ситуацию, в которой используется правило типа

если (A), то (B).

Предположим, что никакие другие правила не имеют отношения к данной ситуации. Разберемся, когда можно сделать вывод, что B истинно.

Где возникает неопределенность? В системах вывода она бывает двух видов. Так, неопределенность возникает при попытке количественно оценить, насколько мы уверены, что предыдущее условие истинно. Например, если степень уверенности того, что A истинно, составляет только 90 %, то какое значение тогда примет B?

Другой вариант – неопределенность в самой импликации. Например, можно сказать, что в большинстве случаев, но не всегда, если есть A, то есть

также и *B*. Должно быть числовое выражение этого факта (скажем, на 95 % уверены, что, имея *A*, имеем и *B*).

Как можно все эти отношения выразить в терминах вероятности? Если бы была абсолютная гарантия, что предшествующее событие *A* истинно, то можно записать:

$$p(A) = 1{,}0.$$

Когда полной определенности нет, установленное значение вероятности отражает эту информацию таким образом

$$p(A) = 0{,}9.$$

При неопределенности второго типа утверждение с вероятностью в 95 %, что будет *B*, если есть *A*, записывается в форме

$$p(B|A) = 0{,}95.$$

Здесь использована условная вероятность. Эта формулировка достаточно ясна сама по себе, но она не дает никакой информации о том, может ли быть *B*, если нет *A*. В ряде случаев такая возможность есть, и тогда нужно получить значение *B* при отсутствии *A*:

Рассмотрим теперь типичную проблему, включающую простую импликацию. Импликация представляет собой выражение типа:

если (*A*), то (*B*).

В другой ситуации для получения заключения могут присутствовать две посылки

если (A и B), то (C),

условная вероятность снова дает все необходимые сведения для рассуждения в этой ситуации.

К сожалению, при постановке задачи может указываться недостаточное количество данных, которое не позволяет прийти к точному ответу.

Вот типичная проблема. Например, вы уверены на 95 %, что A и B влекут за собой C. При этом с вероятностью в 80 % известно, что A истинно, и с вероятностью в 70 % – что B истинно. Какова вероятность C?

Известно следующее:

$$p(A) = 0{,}8;$$
$$p(B) = 0{,}7;$$
$$p(C|A \text{ и } B) = 0{,}95;$$
$$p(C) \text{ – искомый результат.}$$

При использовании правила композиции получаем для C

$$p(C) = p(C|A \text{ и } B) + p(C| \text{ не}(A \text{ и } B))\, p(\text{не}(A \text{ и } B));$$

$$p(C) = 0{,}95\, p(A \text{ и } B) + p(C|\text{не}(A \text{ и } B))\, (1 - p(A \text{ и } B)).$$

Теперь имеем две неизвестные величины вместо одной.

При таких рассуждениях решающее значение приобретает соотношение между $p(A)$ и $p(B)$. К сожалению, как станет ясно из дальнейшего, указанное соотношение нельзя вычислить. Невозможно получить $p(A$ и $B)$ на основе вероятностей компонент, хотя эксперты – не математики часто формулируют свои правила так, как будто подобная связь существует.

Приведенный выше пример показывает, что простые вероятности компонентов дают мало информации о возможности совместного наступления собы-

тия. Однако они указывают интервал, в который эта вероятность обязательно попадет.

Если на этой основе нужно сделать вывод, что невозможно получить дополнительную информацию, тогда, вероятно, наилучшими окажутся следующие предположения

$$p(C \text{ и не } (A \text{ и } B)) = 0,$$

и $p(A$ и $B)$ имеет среднее значение в допустимом интервале.

Однако это уже эвристические рассуждения, а не математически точные.

Существует еще один тип импликации, обычно встречаемый в системах вывода

если (A или B), то (C).

Если параметры выражения не определены, то что можно сказать о C?

Типичную задачу можно сформулировать следующим образом:

Если истинно только A, в 70 % случаев C также будет истинным.

Если истинно только B, в 80 % случаев C также будет истинным.

Если истинно и A, и B, в 95 % случаев C также будет истинным.

Кроме того, в данной конкретной ситуации на 80 % уверены, что A истинно, и на 80 % уверены, что B истинно.

Какова вероятность истинности C?

Сформулируем задачу в терминах теории вероятности.

$$p(C|A \text{ и (не } B)) = 0,7;$$
$$p(C|(\text{не } A) \text{ и } B) = 0,8;$$
$$p(C|A \text{ и } B)) = 0,95;$$
$$p(A) = 0,8;$$
$$p(B) = 0,8.$$

Запишем выражения для $p(C)$, перечислив все условия, когда может произойти событие C (обращаемся к правилу композиции)

$$p(C) = p(C|\ A \text{ и } B)p(A \text{ и } B) +$$
$$+ p(C \mid A \text{ и (не } B))p(A \text{ и (не } B)) +$$
$$+ p(C \mid (\text{не } A) \text{ и } B)\ p((\text{не } A) \text{ и B}) +$$
$$+ p(C \mid (\text{не } A) \text{ и (не } B))\ p((\text{не } A) \text{ и (не } B)).$$

Все остальные условия включают интервалы, вычисляемые с помощью лежащих в их основе конкретных вероятностей, как делалось в предыдущей задаче.

Если на основе этих формальных выкладок все-таки необходимо сделать вывод, то остается лишь надеяться, что следующее равенство справедливо

$$p(C \mid (\text{не } A) \text{ и (не } B)) = 0.$$

Есть еще четыре не известные вероятности, для которых можно вычислить интервалы. Наилучшей оценкой такой величины разумно считать середину допустимого для нее интервала.

Таким образом, строгое рассуждение, построенное на основе вероятностей, является трудным и неудобным.

По этой причине многие экспертные системы не используют рассуждения на основе условной вероятности и выполняют лишь грубые оценки условных вероятностей или вырабатывают некоторые специальные схемы, отображающие то, что мог бы сделать эксперт.

Можно создать много разных схем приближенного рассуждения. Ниже рассмотрим одну из схем механизма рассуждений медицинской системы вывода.

Рассмотрим простой тип импликации:

если (E), то (C).

Эффективный способ решения – присвоить *коэффициент определенности* как посылке, так и всей импликации. Тогда можно совместно использовать эти две величины для вычисления коэффициентов определенности всего заключения.

Что такое *коэффициент определенности*? Его часто применяют вместо понятия вероятности. Если обозначить коэффициент определенности как *ct*, а вероятность как *p*, то в простейшем случае получим:

коэффициент определенности посылки – $ct(E) \sim p(E)$,
коэффициент определенности импликации – $ct(C) \sim p(C|E)$.

Таким образом, коэффициент определенности события приблизительно эквивалентен вероятности того, что посылка является истинной. Коэффициент определенности импликации сходен с условной вероятностью заключения, полученного при истинности посылки.

Обычное правило комбинирования, позволяющее вычислить коэффициент определенности заключения в случае, когда известен коэффициент определенности посылки, лежащей в его основе, и связи в импликации, записывается так

$$ct(\text{заключение}) = ct(\text{посылка}) \times ct(\text{импликация}).$$

Например, верим в истинность посылки с вероятностью 0,8. Верим и в то, что лежащая в основе импликации схема выполняется в большинстве случаев, но не всегда. Поэтому приписываем ей коэффициент определенности 0,9. Тогда коэффициент определенности заключения в такой ситуации

$$ct(\text{заключение}) = 0{,}8 \times 0{,}9 = 0{,}72.$$

Прежде всего нужно суметь оценить коэффициенты определенности посылок. Будем называть посылкой все логическое выражение в правиле между “если” и “то”. За исключением случаев простой импликации, это выражение состоит из атомарных посылок, каждая из которых имеет свой коэффициент определенности. Они могут быть связаны между собой логическими операциями, например:

$$\text{если}(e1 \text{ или } (e2 \text{ и } e3)), \text{ то } (c)$$

или

$$\text{если } (e1 \text{ и } e2 \text{ и } ((\text{не } e3) \text{ или } e4)), \text{ то } (c).$$

Очевидно, требуется некоторый способ оценки коэффициентов определенности этих сложных форм в понятияхих отдельных компонент.

Подход заключается в том, чтобы отбросить все сложные выражения и считать все правила простыми. Такое ограничение, тем не менее сохраняет структуры правил, которые являются достаточно информативными для большинства целей. Есть несколько тривиальных процедур для сведения коэффициентов определенности простых логических комбинаций в одно число.

Простейшей логической комбинацией является конъюнкция между двумя элементарными свидетельствами

$$\text{если } (e1 \text{ и } e2), \text{ то } (c).$$

Коэффициент определенности посылки равен коэффициенту определенности наименее надежной из посылок, т.е.

$$ct(e\text{l} \text{ и } e2) = \min (ct(e\text{l}), ct(e2)).$$

Другой простой формой является правило, в котором используется дизъ-

юнкция, связывающая две части свидетельств:

если (*e*1 или *e*2), то (*c*).

Общее правило комбинирования, по которому вычисляется коэффициент определенности посылки, заключается в том, что коэффициент определенности дизъюнкции равен коэффициенту определенности ее сильнейшей части, т.е.

$$ct(e1 \text{ или } e2) = \max(ct(e1), ct(e2)).$$

Хотя правила иногда и записываются с помощью дизъюнкции, если есть выбор, то принято разбивать дизъюнкцию на две части, например:

если (*e*1), то (*c*),
если (*e*2), то (*c*).

Использование двух правил вместо дизъюнкции позволяет более отчетливо увидеть ситуацию, но если необходимо придерживаться этого соображения, то нужен механизм, определяющий коэффициент определенности заключения при поддержке двух правил.

Рассмотрим ситуацию, когда используются два правила и оба они поддерживают одно и то же заключение, например:

Правило 1: если (*e*1), то (*c*), *ct*(заключение) = 0,9.

Правило 2: если (*e*2), то (*c*), *ct*(заключение) = 0,8.

Допустим, обе посылки верны, и мы вычислили вероятность заключения для каждого правила по отдельности.

Если использовалось правило 1, то коэффициент определенности в заключении окажется равным 0,9. Ясно, что, имея еще и правило 2, получим

больший коэффициент определенности, но какова будет его величина?

Предположим, что переменная *ctotal* представляет общий коэффициент определенности заключения, полученный использованием всех поддерживающих его правил. Можно предложить много различных комбинаций процедур. Например, хорошо действует простой и эффективный механизм вычисления общего коэффициента определенности заключения:

ctotal = коэффициент определенности из правила 1 +
+ коэффициент определенности из правила 2 –
– (коэффициент определенности из правила 1) ×
× (коэффициент определенности из правила 2).

Подставив числа, заданные в примере, получим

$$ctolal = 0{,}9 + 0{,}8 - (0{,}9)(0{,}8) = 0{,}98.$$

Рассмотренный принцип можно распространить на случай *n* правил, поддерживающих одно заключение, где для каждого правила существует своя вероятность.

Например, есть три правила со следующими коэффициентами определенности:

Правило 1: если (*e*l), то (*c*), *ct*(заключение) = *ct*l.

Правило 2: если (*e*2), то (*c*), *ct*(заключение) = *ct*2.

Правило 3: если (*e*3), то (*c*), *ct*(заключение) = *ct*3.

Совокупный коэффициент определенности заключения с учетом всей возможной поддержки может быть вычислен как

$$ctotal = ct1 + ct2 + ct3 - ct1 \times ct2 - ct2 \times ct3 - ct1 \times ct3 + ct1 \times ct2 \times ct3.$$

Таким образом, коэффициент определенности – это приблизительные рассуждения.

5.3.Метод дополнения

Механизм дополнения – это другой способ вычисления коэффициента определенности заключения, поддерживаемого несколькими правилами импликации. Он используется в случае, когда сведения о разрешенных к применению правилах поступают последовательно, а не одновременно. Например, если система задает пользователю вопросы, то применение новых правил будет происходить по очереди [10].

Рассмотрим пример. Допустим, известно, что заключение поддерживается двумя правилами со следующими коэффициентами определенности.

Правило 1: если (*e*1), то (*c*), *ct*(заключение) = *ct*l.

Правило 2: если (*e*2), то (*c*), *ct*(заключение) = *ct*2.

При применении двух правил совокупный коэффициент определенности

$$ctotal = ct1 + ct2 - ct1 \times ct2.$$

Теперь предположим, что появилось третье правило, поддерживающее то же заключение:

Правило 3: если (*e*3), то (*c*), *ct*(заключение) = *ct*3.

Если все, что получено из предыдущего исследования, входит в переменную *ctotal*, и если считается, что *ct*3 может войти в рассуждения на общих основаниях, то можно использовать стратегию дополнения для формирования измененной оценки коэффициента определенности заключения

$$cnewtotal = ct3 + ctotal - ct3 \times ctotal.$$

В любом случае при использовании механизма дополнения порядок поступления правил, поддерживающих заключение, не имеет значения.

Можно объединять коэффициенты определенности из поддерживающих импликаций последовательно по мере их поступления или сохранять информацию, а затем использовать ее всю сразу – результат от этого не меняется.

Практически сеть рассуждения меняется, как только поступают новые сведения. Поэтому сохранять нужно лишь совокупный коэффициент определенности для каждого заключения, что обеспечивает наиболее экономный способ поддержки информационного обеспечения ЭС.

5.4. Биполярные схемы для коэффициентов определенности

В системах, основанных на приближенных рассуждениях для численного выражения определенности используется интервал от –1 до + 1, так что это не может быть вероятностью. Границы интервала обозначают следующее: + 1 – система в чем-то полностью определена, 0 – у системы нет знаний об обсуждаемой величине, –1 – высказанная гипотетическая посылка или заключение абсолютно неверно. Промежуточные величины отражают степень доверия или недоверия к указанным ситуациям [10].

Все описанные процедуры рассуждений применимы и для коэффициентов определенности, задаваемых в этих более широких границах.

Полная реализация идеи биполярных коэффициентов определенности требует сделать два обобщения.

Первое, если в правиле есть отрицание, например:

если (*e*1 и (не *e*2)), то (*c*),

то нужно считать (не *e*2) новым атомарным утверждением, например, *e*3. Для вычисления же коэффициента определенности (не *e*2) достаточно просто поменять знак:

$$c1(\text{не } e) = -ct(e).$$

Другое – это правило получения коэффициентов определённостей в условиях поддержки двумя правилами одного и того же заключения:

– если оба коэффициента определенности положительны, то

$$ctotal = ct1 + ct2 - ct1 \times ct2;$$

– если оба коэффициента определенности отрицательны, то

$$ctotol = ct1 + ct2 + ct1 \times ct2.$$

Когда отрицателен один коэффициент, то:

$$ctotal = (ct1 + ct2)/(1 - \min(\text{abs}(ct1), \text{abs}(ct2))).$$

В том случае, когда одна определенность равна +1, а другая – 1,

$$ctotal = 0.$$

Когда два правила с небольшими коэффициентами определенности поддерживают одно заключение, коэффициент определенности заключения возрастает. Если же знаки не совпадают, то результат определяется сильнейшим, но влияние его несколько ослабляется. Применение биполярных коэффициентов определенности может привести к нереальным результатам, если правила

сформулированы неточно. При работе с одним правилом вывода всегда используется соотношение:

$$ct(\text{заключение}) \sim ct(\text{посылка}) \times ct(\text{импликация}).$$

Все правила попадают в одну из двух очень важных категорий.

Правила первой категории будем называть *обратимыми.* Правило считается обратимым, если при добавлении отрицания не и к условию, и к выводу оно не теряет смысл. Одной из характеристик такого правила является его применимость к любому вероятностному значению, которое может быть связано с посылкой.

Правила второй категории считаются *не обратимыми*. Эти правила работают только при положительных значениях посылки. Если же ее значение отрицательно, правило применять нельзя.

Любое правило может быть отображено графически (рис. 5.2). Совокупность правил называется *сетью вывода*.

Рассмотрим пример, иллюстрирующий распространение коэффициентов определенности в сети. Узлы в основании дерева представляют условия из внешнего мира, о которых система должна задавать вопросы. Внутренние узлы отображают заключения. Коэффициенты определенности внутренних узлов полностью зависят от процесса рассуждения, правил импликации и свидетельств, полученных из внешнего мира путем запросов. Число у каждого узла соответствует коэффициенту определенности. До начала рассуждений ничего не известно о внутренних узлах, поэтому они все имеют коэффициенты определенности, равные нулю.

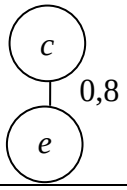
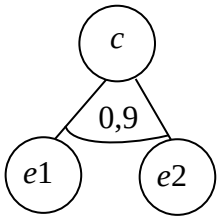
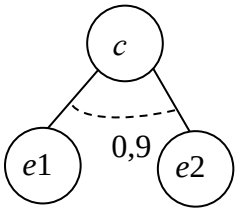
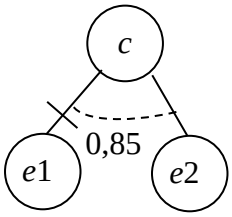
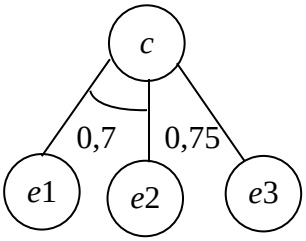
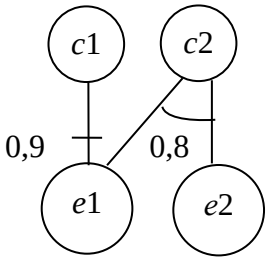
Простая импликация	
Если *e*, то *c*, *ct* = 0,8.	
Импликация AND	
Если (*e*1 *and e*2), то *c*, *ct* = 0,9.	
Импликация OR	
Если (*e*1 *or e*2), то *c*, *ct* = 0,9.	
Импликация с отрицанием NOT	
Если ((*not e*1) *or e*2) то *c*, *ct* = 0,85.	
Несколько правил в поддержку одного заключения	
Если (*e*1 *and e*2) *or e*2), то *c*, *ct* = 0,7. Если (*e*3), то *c*, *ct* = 0,75.	
Одно свидетельство, используемое в двух правилах	
Если (*not e*1), то *c*1, *ct* = 0,9. Если (*e*1 *and e*2), то *c*2, *ct* = 0,8.	

Рис. 5.2. Графическое представление правил

Сеть вывода (рис. 5.3) предполагает следующие правила.

Если (*e*1), то (*c*1), *ct* = 0,8 (*nrev*).

Если (*e*2), то (*c*2), *ct* = 0,9 (*rev*).

Если (*e*3), то (*c*2), *ct* = 0.7 (*rev*).

Если (*e*4), то (*c*3), *ct* = 0,6 (*nrev*).

Если (не *e*5), то (*c*3), *ct* = 0,5 (*nrev*).

Если (*c*2 и *c*3), то (*c*4), *ct* = 0,9 (*rev*).

Если (*c*1 или *c*4), то (*c*5), *ct* = 0,8 (*nrev*).

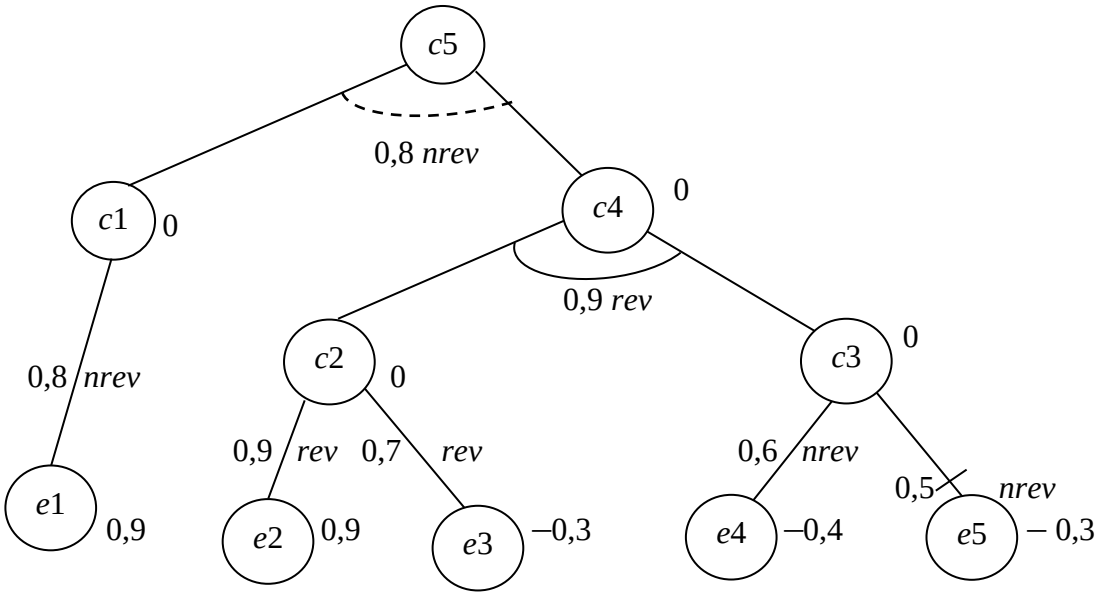


Рис. 5.3. Пример сети вывода для проведения рассуждений с заданными начальными условиями

Рядом с коэффициентом определенности импликации *rev* (обратимо) или *nrev* (необратимо), что обозначает, будет ли импликация использоваться как обратимое или как необратимое правило. Вычисление коэффициента определенности посылки может потребовать выполнения нескольких шагов: могут добавляться логические операции И, ИЛИ, НЕ. В каждом конкретном случае, пока не будет закончена вся эта предварительная работа, нельзя с уверенностью сказать, применимо ли правило.

Коэффициент определенности *c*1 может быть вычислен следующим обра-

зом

$$ct(\text{заключение } c1) = 0{,}8 \times 0{,}9 = 0{,}72.$$

Это простая необратимая импликация, но поскольку коэффициент определенности посылки позитивен, правило можно применять.

При вычисления коэффициента определенности $c2$ оба правила используются без ограничений, так как они обратимы. Правило слева даст оценку коэффициента определенности $c2$:

$$ct(\text{заключение } c2) = 0{,}9 \times 0{,}9 = 0{,}81.$$

Правило справа даст иную оценку:

$$ct(\text{заключение } c2) = -0{,}3 \times 0{,}7 = -0{,}21.$$

Здесь два поддерживающих правила, дающих оценку коэффициента определенности с противоположными знаками, поэтому для окончательного ответа объединим эти оценки:

$$ct(\text{заключение } c2) = (0{,}81 + (-0{,}21))/(1 - 0{,}21) = 0{,}74.$$

Для $c3$ опять два правила. Правило, связанное с левым поддеревом, не применяется, так как оно необратимо, и коэффициент определенности посылки отрицателен. Правило, связанное с правым поддеревом, есть простая импликация. Она необратима и содержит отрицательную посылку. Правило утверждает:

$$\text{если (не } e5\text{), то (с3)}, \quad ct = 0{,}5 \ (nrev).$$

Коэффициент определенности $e5$ равен – 0,3. Так как он негативен, то коэффициент определенности посылки в правиле равен 0,3. Правило необратимо, но посылка находится в допустимом интервале. Используя процедуру, предна-

значенную для простой импликации, найдем для *c*3:

$$ct(\text{заключение } c3) = 0{,}3 \times 0{,}5 = 0{,}15.$$

Импликация, поддерживающая *c*4, включает конъюнкцию посылок. Коэффициент определенности посылки равен:

$$ct(\text{свидетельства}) = \min(0{,}15;\ 0{,}74) = 0{,}15.$$

Поскольку правило обратимо, можно использовать посылку в любом интервале определенности. Используя этот результат, вычислим коэффициент определенности для *c*4:

$$ct\ (\text{заключение } c4) = 0{,}15 \times 0{,}9 = 0{,}13.$$

Теперь прошли вверх по дереву до того места, где можно судить об узле верхнего уровня. Здесь задействовано одно правило, в котором посылки разделены с помощью ИЛИ, поэтому:

$$ct\ (\text{свидетельства}) = \max\ (0{,}72;0{,}13) = 0{,}72.$$

Правило необратимо, но коэффициент определенности посылки позитивен, так что можем двигаться дальше.

Последнее звено в нашей цепи рассуждении – коэффициент определенности для узла высшего уровня – вычисляется по формуле:

$$ct\ (\text{заключение } c5) = 0{,}72 \times 0{,}8 = 0{,}58.$$

Ниже показан результат всех рассуждении, проводимых в сети, использующей подходящее свидетельство (рис. 5.4).

Обычно сети вывода требуются в ситуации, когда при наличии нескольких конкурирующих гипотез экспертная система пытается сделать выбор и по-

родить информацию. Например, в медицинских системах сеть выбора может использоваться при установлении причин болей у пациента: аппендицит ли это, рак, какая-нибудь инфекция, вызвавшая воспаление лимфатических узлов, или, возможно, просто расстройство желудка.

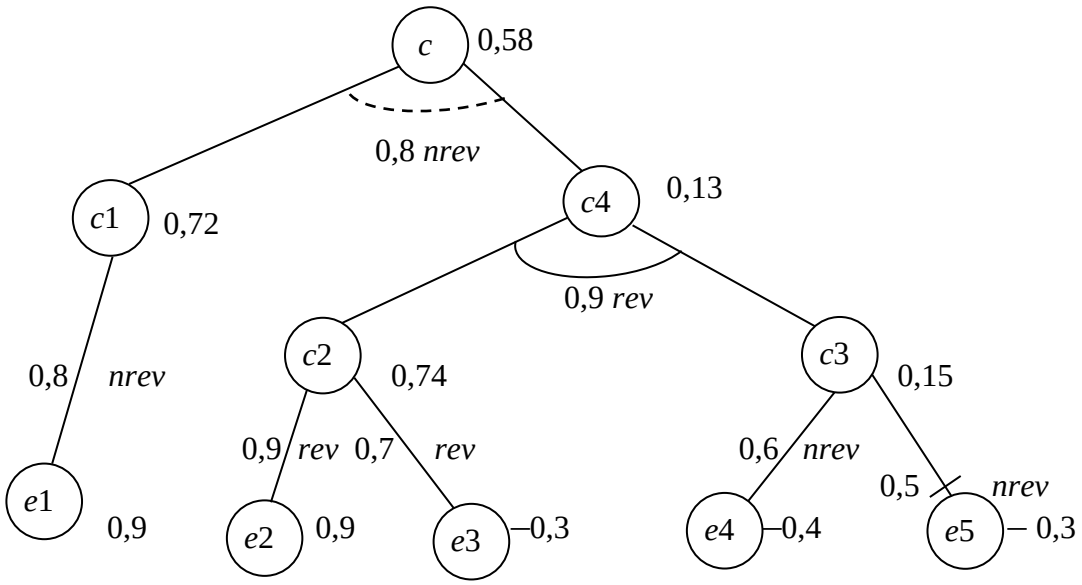


Рис. 5.4. Пример сети с вычисленными коэффициентами

5.6. Теория свидетельств Демпстера–Шефера

Подход, принятый в теории Демпстера–Шефера (ТДШ) отличается от байесовского подхода и коэффициентов уверенности тем, что, во-первых, здесь используется не точечная оценка уверенности (коэффициент уверенности), а интервальная оценка. Такая оценка характеризуется нижней и верхней границами, что более надежно. Во-вторых, ТДШ позволяет исключить взаимосвязь между неопределенностью (неполнотой знаний) и недоверием, которая свойственна байесовскому подходу [1, 7, 8].

В рамках ТДШ множеству высказываний A приписывается диапазон значений $[Bl(A), P1(A)]$, в котором находятся степени доверия (правдоподобия) каждого из высказываний. Здесь $B1(A)$ – степень доверия к множеству высказываний, изменяющая свои значения от 0 (нет свидетельств в пользу A) до 1 (множество высказываний A истинно); $P1(A)$ – степень правдоподобия множества высказываний A, определяемая с

помощью формулы: $Pl(A) = 1 - Bl(\text{not}A)$.

Предположим, что существуют две конкурирующие гипотезы h_1 и h_2. При отсутствии информации, поддерживающей эти гипотезы, мера доверия и правдоподобия каждой их них принадлежат отрезку [0; 1]. По мере накопления эти интервалы будут уменьшатся, а доверие к гипотезам – увеличиваться. В теории Демпстера–Шефера неопределенность знаний представляется с помощью некоторого множества X. Элементы этого множества соответствуют возможным фактам или заключениям. Неопределенность состоит в том, что заранее неизвестно, какое из возможных значений примет факт или заключение $x \in X$.

Для характеристики степени определенности в ТДШ вводится некоторая единичная мера уверенности (ее называют также единичной массой уверенности), которая распределяется между элементами X. При этом, если вся масса (степень) уверенности приходится на один элемент $x \in X$, то никакой неопределенности нет. Неопределенность возникает, когда степень уверенности распределяется между несколькими элементами $x_i \in X$.

В общем случае распределение степени уверенности задается функцией $m(A)$, обладающей следующими свойствами:

$$m(\varnothing)=0,$$
$$\sum m(A)=1.$$

Здесь A – множество, образованное из подмножеств X, которым назначены соответствующие степени уверенности; $m(A)$ – функция, которая задает отображение A на интервал [0, 1]. Множество с количеством элементов N имеет точно 2^N подмножеств, включая самого себя, например при $N = 3$, имеем:

$$A = \{\varnothing, \{x1\}, \{x2\}, \{x3\}, \{x1, x2\}, \{x2, x3\}, \{x1, x3\}, \{x1, x2, x3\}\},$$

а распределение масс уверенности задаётся функцией $m(A)$, характеризуемой множеством значений $m(A)$.

A состоит из подмножеств. Обозначим каждое такое подмножество через A_i.

Степень доверия к высказываниям, соответствующим подмножеству A_i, может быть вычислена по формуле

$$Bl(\{A_i\}) = \sum_{A_j \in A_i} m_{A_j}(A).$$

Степень правдоподобия подмножества A_i определяется по формуле

$$Pl(A_i) = 1 - Bl(not\ A_i) = 1 - \sum_{A_j \in A_i} m_{A_j}(A_j).$$

Величины $Bl(A_i)$ и $Pl(A_i)$ имеют простую интерпретацию: $Bl(A_i)$ представляет общую массу уверенности, которая остается, если из *X* удалить все элементы, не ассоциируемые с A_i. $Pl(A_i)$ представляет максимальную массу уверенности, которую можно получить, если сдвинуть свободные массы к элементам множества A_i. Причем $Bl(A_i) \le Pl(A_i)$.

Иными словами, $Bl(A_i)$ представляет нижнюю границу доверия к A_i, а $Pl(A_i)$ – верхнюю.

Важнейшим элементом ТДШ является правило комбинации свидетельств

$$m(A_k) = \frac{\sum_{Ai1 \cap A2\,j} m_1(A_{1i}) m_2(A_{2\,j})}{1 - \sum_{Ai2 \cap A2\,j = \varphi} m_1(A_{1i}) m_2(A_{2\,j})}.$$

Сумма в числителе правила распространяется на множество $A_k = A_{1i} \cap A_{2\,j}$.

Правило является эвристическим и позволяет осуществлять распределение степеней доверия в ходе вывода.

Например, мерой доверия $m_n(Z)$ гипотезе *Z* для *n* = 3 источников свидетельств считается сумма произведений гипотетических мер доверия $m_1(X)$ и $m_2(Y)$, совмест-

ное вхождение которых поддерживает Z, т.е. $X \cap Y = Z$.

Знаменатель в правиле Демпстера допускает пустое пересечение X и Y, а сумма мер доверия должна быть нормализована.

Рассмотрим применение правила Демпстера для задачи медицинской диагностики, описанное в [10].

Предположим, что есть область Q, содержащая четыре гипотезы:

1. Пациент был без сознания (C).

2. У него был грипп (F).

3. Возможно это мигрень (H).

4. Возможно это менингит (M).

Необходимо связать меры доверия со множествами гипотез в рамках Q. Например, лихорадка свидетельствует в пользу версий $\{C, F, M\}$. Так как элементы Q трактуются как взаимоисключающие гипотезы, подтверждение одной из них может влиять на достоверность других.

Пусть есть свидетельство, что у пациента лихорадка (температура). Оно поддерживает версии $\{C, F, M\}$ с вероятностью 0,6. Назовем это первой мерой доверия m_1. Если это всего лишь гипотеза, то

$$m_1\{C, F, M\} = 0{,}6,$$

где $m_1\{Q\} = 0{,}4$, остаток $(1 - 0{,}6)$ определяет оставшуюся часть распределения достоверности, т.е. все другие возможные меры доверия Q, а не достоверность дополнения $\{C, F, M\}$.

Затем были получен факт о новом проявлении болезни – у пациента рвота, которая свидетельствует о $\{C, F, H\}$ со степенью доверия 0,7.

Пусть это будет мера доверия свидетельства m_2. Тогда имеем

$$m_2\{C, F, H\} = 0{,}7, \quad \text{и} \quad m_2\{Q\} = 0{,}3.$$

Получаем таким образом множество X – набор подмножеств Q на котором m_1 принимает ненулевые значения, и Y – набор подмножеств Q на котором

m_2 принимает ненулевые значения.

Применим правило Демпстера для определения объединенной меры доверия m_3: перемножим X и Y.

Знаменатель пока равен 1, т.к. пока не существует пустых множеств $X \cap Y$. Результат вычислений в табл. 5.1.

Таблица 5.1 – Применение правила для объединенияm_1 и m_2

m_1	m_2	m_3
$m_1\{\boldsymbol{C},\boldsymbol{F},M\}$=0,6	$m_2\{\boldsymbol{C},\boldsymbol{F},H\}$=0,7	$m_2\{\boldsymbol{C},\boldsymbol{F}\}$=0,42
$m_1\{Q\}$=0,4	$m_2\{C,F,H\}$=0,7	$m_2\{C,F,H\}$=0,28
$m_1\{C,F,M\}$=0,6	$m_2\{Q\}$=0,3	$m_2\{C,F,M\}$=0,18
$m_1\{\boldsymbol{Q}\}$=0,4	$m_2\{\boldsymbol{Q}\}$=0,3	$m_3\{\boldsymbol{Q}\}$=0,12

Обратите внимание на рассуждения и группировки гипотез.

Четыре множества столбца m_3 представляют собой все возможные пересечения X и Y.

Этих данных явно недостаточно для установки диагноза, что и отражают полученные числа.

Добавим данные лабораторных анализов, которые свидетельствует в пользу менингита

$$m_4\{M\} = 0{,}8 \text{ и } m_4\{Q\} = 0{,}2.$$

Применим еще раз правило Демпстера для определения объединенной меры доверия m_5. Результат вычислений дан в таблице 5.2.

Так как $m_5\{M\}$ получается в нескольких случаях, то общая вероятность

$$m_5\{M\} = (0{,}144 + 0{,}096) = 0{,}240.$$

В результате пересечения нескольких пар множеств получается пустое множество {}, значит знаменатель в уравнение Демпстера нужно считать, как

$$(1 – (0{,}336 + 0{,}224)) = 0{,}44.$$

Таблица 5.2 – Применение правила для объединения m_3 и m_4

m_3	m_4	m_5
$m_2\{C,F\}=0{,}42$	$m_4\{M\}=0{,}8$	$m_5\{\}=0{,}336$
$m_3\{Q\}=0{,}12$	$m_4\{M\}=0{,}8$	$m_5\{M\}=0{,}096$
$m_2\{C,F\}=0{,}42$	$m_4\{Q\}=0{,}2$	$m_5\{C,F\}=0{,}084$
$m_3\{Q\}=0{,}2$	$m_4\{Q\}=0{,}2$	$m_5\{Q\}=0{,}024$
$m_2\{C,F,H\}=0{,}28$	$m_4\{M\}=0{,}8$	$m_5\{\}=0{,}224$
$m_2\{C,F,M\}=0{,}18$	$m_4\{M\}=0{,}8$	$m_5\{M\}=0{,}44$
$m_2\{C,F,H\}=0{,}28$	$m_4\{Q\}=0{,}2$	$m_5\{C,F,H\}=0{,}056$
$m_2\{C,F,M\}=0{,}18$	$m_4\{Q\}=0{,}2$	$m_5\{C,F,M\}=0{,}036$

Окончательные значения меры доверия имеют вид:

$$m_5\{M\} = 0{,}240/0{,}44 = 0{,}545,$$
$$m_5\{C, F\} = 0{,}084/0{,}44 = 0{,}191,$$
$$m_5\{C, F, H\} = 0{,}056/0{,}44 = 0{,}127,$$
$$m_5\{C, F, M\} = 0{,}036/0{,}44 = 0{,}82,$$
$$m_5\{Q\} = 0{,}024/0{,}44 = 0{,}055,$$
$$m_5\{\} = 0{,}336 + 0{,}224 = 0{,}56.$$

Высокая достоверность пустого множества $m_5\{\} = 0{,}56$ означает существование конфликта свидетельств на множестве мер доверия m_j т.к. в примере даны некорректные с точки зрения медицины данные.

При существовании больших множеств гипотез вычисление мер доверия может оказаться громоздким, но все же значительно меньше чем при использовании теоремы Байеса.

Правило Демстера – пример рассуждений субъективных вероятностей, в отличие от объективных вероятностей Байеса.

Контрольные вопросы

1. Допустим, в задаче на все вопросы получены ответы ДА или НЕТ. Сосредоточим внимание на двух конкретных вопросах, которые назовем “вопрос *A*” и ”вопрос *B*”. Потребители, как правило, отвечают на эти вопросы одинаково, т.е. те, кто ответил ДА на вопрос *A*, вероятно, так же ответят и на вопрос *B*, а те, кто ответил НЕТ на вопрос *A*, вероятно, так же ответят и на вопрос *B*.

Пусть $p(A)$ – вероятность ответа ДА на вопрос *A*. Известно, что $p(A) = 0{,}6$и ответы на оба вопроса значительно коррелируют друг с другом, поэтому:

$$p(B|A) = 0{,}9; \quad p(A|B) = 0{,}8; \quad p(\text{не } B|\text{не } A) = 0{,}85.$$

Вычислите следующие вероятности: $p(B)$; $p(\text{не } A|\text{не } B)$; $p(A \text{ и } B)$.

Какова вероятность того, что в любом испытании ответы на вопросы *A* и *B* будут одинаковы?

2. Укажите, будет ли обратимым каждое из следующих правил:

Если команда делает хорошие броски, то она хорошо играет.

Если человек жив, то в нем течет кровь.

Если были ранние заморозки, то урожай персиков будет плохим.

3. В чем отличие ТДШ от остальных подходов, приведенных в работе?

4. Как изменяется доверие к гипотезам по мере решения?

5. Как вычисляется оценка нижней и верхней границы ТДШ?

6. Как вычисляется степень доверия к высказываниям, соответствующим подмножеству A_i?

7. Как вычисляется степень правдоподобия подмножества A_i?

8. Как вычисляется знаменатель в правиле Демпстера?

Подвёл ты меня, Боролгин. А ведь я все деньги на тебя поставил.
…<тут Боролгин сокрушается> ...
Не горюй, Боролгин. Я ещё и на орков поставил.

“Властелин колец” Дж. Р.Р. Толкин

6.ВВЕДЕНИЕ В ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

6.1. Области применения искусственных нейронных сетей

Интеллектуальные системы на основе искусственных нейронных сетей (ИНС) позволяют с успехом решать проблемы распознавания образов, выполнения прогнозов, оптимизации, ассоциативной памяти и управления. Известны и иные, более традиционные подходы к решению этих проблем, однако они не обладают необходимой гибкостью за пределами ограниченных условий. ИНС дают многообещающие альтернативные решения, и многие приложения выигрывают от их использования [14-18].

Представим некоторые проблемы, решаемые в контексте ИНС и представляющие интерес для ученых и инженеров.

✓ *Классификация образов.* Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови.

✓ *Кластеризация/категоризация.* При решении задачи кластеризации, которая известна также как классификация образов ”без учителя”, отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на

подобии образов и размещает близкие изображения в один кластер. Известны случаи применения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.

✓ *Аппроксимация функций.* Предположим, что имеется обучающая выборка $((x_1,y_1), (x_2,y_2), ..., (x_n,y_n))$ (пары данных вход-выход), которая генерируется неизвестной функцией $f(x)$, искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции $f(x)$. Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

✓ *Предсказание/прогноз.* Пусть заданы n дискретных отсчетов $\{y(t_1), y(t_2), ..., y(t_n)\}$ в последовательные моменты времени $t_1, t_2,..., t_n$. Задача состоит в предсказании значения $y(t_{n+1})$ в некоторый будущий момент времени t_{n+1}. Предсказание/прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза.

✓ *Оптимизация.* Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Задача коммивояжера, относящаяся к классу NP-полных, является классическим примером задачи оптимизации.

✓ *Память, адресуемая по содержанию.* В модели вычислений фон Неймана обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Ассоциативная память, или память, адресуемая по содержанию, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании мультимедийных информационных баз данных.

✓ *Управление.* Рассмотрим динамическую систему, заданную совокупностью $\{u(t), y(t)\}$, где $u(t)$ является входным управляющим воздействием, а $y(t)$ – выходом системы в момент времени t. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия $u(t)$, при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление двигателем.

ИНС может рассматриваться как направленный граф со взвешенными связями, в котором искусственные нейроны являются узлами. По архитектуре связей ИНС могут быть сгруппированы в два класса: сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями.

6.2. Алгоритмы обучения искусственных нейронных сетей

Способность к обучению является фундаментальным свойством мозга. В контексте ИНС процесс обучения может рассматриваться как настройка архитектуры сети и весов связей для эффективного выполнения специальной задачи. Обычно нейронная сеть должна настроить веса связей по имеющейся обучающей выборке. Функционирование сети улучшается по мере итеративной настройки весовых коэффициентов. Свойство сети обучаться на примерах делает их более привлекательными по сравнению с системами, которые следуют определенной системе правил функционирования, сформулированной экспертами.

Существуют три парадигмы обучения: ”с учителем”, ”без учителя” (самообучение) и смешанная.

В первом случае нейронная сеть располагает правильными ответами (выходами сети) на каждый входной пример. Веса настраиваются так, чтобы сеть производила ответы как можно более близкие к известным правильным отве-

там. Усиленный вариант обучения с учителем предполагает, что известна только критическая оценка правильности выхода нейронной сети, но не сами правильные значения выхода.

Обучение без учителя не требует знания правильных ответов на каждый пример обучающей выборки. В этом случае раскрывается внутренняя структура данных или корреляции между образцами в системе данных, что позволяет распределить образцы по категориям.

При смешанном обучении часть весов определяется посредством обучения с учителем, в то время как остальная получается с помощью самообучения.

Теория обучения рассматривает три фундаментальных свойства, связанных с обучением по примерам: *емкость, сложность образцов и вычислительная сложность*.

Под емкостью понимается, сколько образцов может запомнить сеть и какие функции и границы принятия решений могут быть на ней сформированы.

Сложность образцов определяет число обучающих примеров, необходимых для достижения способности сети к обобщению. Слишком малое число примеров может вызвать ”переобученность” сети, когда она хорошо функционирует на примерах обучающей выборки, но плохо – на тестовых примерах, подчиненных тому же статистическому распределению.

Известны*4 основных типа правил обучения*: коррекция по ошибке, машина Больцмана, правило Хебба и обучение методом соревнования.

- *Правило коррекции по ошибке.* При обучении с учителем для каждого входного примера задан желаемый выход d. Реальный выход сети y может не совпадать с желаемым. Принцип коррекции по ошибке при обучении состоит в использовании сигнала ($d–y$) для модификации весов, обеспечивающей постепенное уменьшение ошибки. Обучение имеет место в случае, когда сеть ошибается. Известны модификации этого алгоритма обучения.

- *Обучение Больцмана.* Представляет собой стохастическое правило обучения, которое следует из информационных теоретических и термодинамических принципов. Целью обучения Больцмана является такая настройка весовых коэффициентов, при которой состояния видимых нейронов удовлетворяют

желаемому распределению вероятностей. Обучение Больцмана может рассматриваться как случай коррекции по ошибке, в котором под ошибкой понимается расхождение корреляций состояний в двух режимах.

➢ *Правило Хебба.* Самым старым обучающим правилом является постулат обучения Хебба. Хебб опирался на следующие нейрофизиологические наблюдения: если нейроны с обеих сторон синапса активизируются одновременно и регулярно, то сила синаптической связи возрастает. Важной особенностью этого правила является то, что изменение синаптического веса зависит только от активности нейронов, которые связаны данным синапсом.

➢ *Обучение методом соревнования.* В отличие от обучения Хебба, в котором множество выходных нейронов могут возбуждаться одновременно, при соревновательном обучении выходные нейроны соревнуются между собой за активизацию. Это явление известно как правило “победитель берет все”. Подобное обучение имеет место в биологических нейронных сетях. Обучение посредством соревнования позволяет кластеризовать входные данные: подобные примеры группируются сетью в соответствии с корреляциями и представляются одним элементом. При обучении модифицируются только веса ”победившего” нейрона. Эффект этого правила достигается за счет такого изменения сохраненного в сети образца (вектора весов связей победившего нейрона), при котором он становится чуть ближе к входному примеру.

Каждый алгоритм обучения ориентирован на сеть определенной архитектуры и предназначен для ограниченного класса задач.

Перечислим задачи, для решения которых можно применить НС: идентификация образов (категоризация), аппроксимация, ассоциация, оптимизация, сжатие данных, анализ данных, предсказание и управление.

Общими отличительными чертами нейронных вычислений являются:

- реализация вычислений как процессов параллельного взаимодействия через сигналы между множеством однотипных функциональных элементов;
- распределенное представление информации через систему межнейронных связей;
- решение задач обработки информации с помощью процедуры обучения;

• эффективное использование вычислительных ресурсов – память в моделях нейронных вычислений непосредственно участвует в процессе обработки информации, а взаимодействие между нейронами параллельное;

• стабильное функционирование при изменениях в некоторых структурных элементах модели.

В настоящее время предложен ряд моделей нейронных сетей. Однако общими проблемами, которые должны быть решены при разработке или выборе модели нейронных вычислений, являются следующие.

1. Определение топологии модели.

2. Определение способа представления информации.

3. Выбор функции преобразования, реализуемой искусственным нейроном.

4. Построение процедур обучения.

6.3. Формальные нейроны искусственных нейронных сетей

При моделировании нейронных сетей в качестве искусственных нейронов обычно используется простой процессорный элемент (рис. 6.1) [14-18].

На его входы поступает вектор $X = (x_1, \ldots, x_n)$ входных сигналов, являющихся выходными сигналами других нейронов, а также единичный сигнал смещения. Все входные сигналы, включая и сигнал смещения, умножаются на весовые коэффициенты своих связей и суммируются:

$$S = \sum_{i=1}^{n} x_i w_i + w_0, \qquad (6.1)$$

где S – суммарный входной сигнал; w_i $(i = \overline{1,n})$ – весовые коэффициенты связей входных сигналов $x_1, x_2, \ldots, x_n$; w_0 – весовой коэффициент связи сигнала смещения.

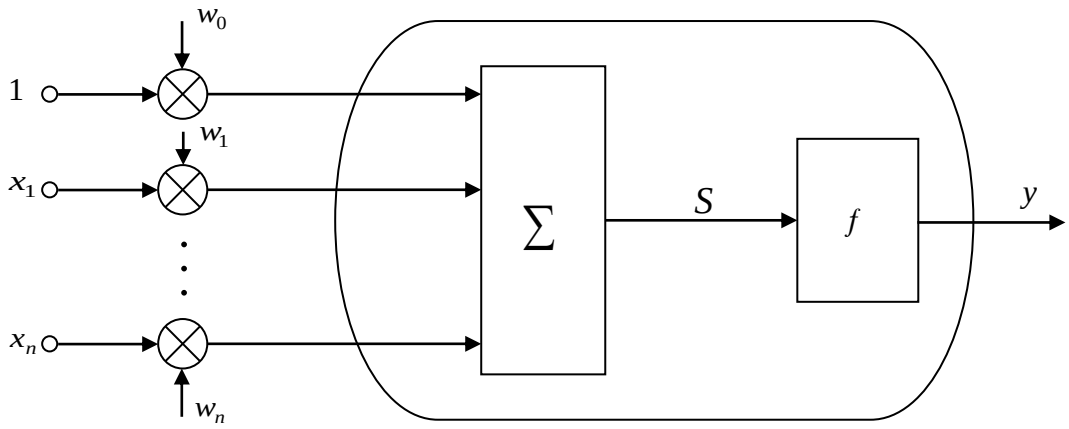


Рис. 6.1. Процессорный элемент, используемый в обычных нейросетях

Полученный сигнал S поступает на вход блока, реализующего функцию активации нейрона f. Типичными функциями активации являются бинарная

$$y = \begin{cases} 1, & \text{если} \quad S > 0, \\ 0, & \text{если} \quad S \le 0, \end{cases} \tag{6.2}$$

или биполярная

$$y = \begin{cases} 1, & \text{если} \quad S > 0, \\ -1, & \text{если} \quad S \le 0. \end{cases} \tag{6.3}$$

Многие авторы при описании модели нейрона используют не сигнал смещения, а порог θ нейрона, что приводит к эквивалентной модели элемента. В этом случае выражения (6.2) и (6.3) принимают соответственно вид:

$$y = \begin{cases} 1, & \text{если} \quad S > \theta, \\ 0, & \text{если} \quad S \le \theta, \end{cases} \tag{6.4}$$

$$y = \begin{cases} 1, & \text{если} \quad S > \theta, \\ -1, & \text{если} \quad S \le \theta, \end{cases} \tag{6.5}$$

где

$$S = \sum_{i=1}^{n} w_i \, x_i \, . \tag{6.6}$$

Графическое изображение бинарной и биполярной функций активации для этого случая представлено на рис. 6.2, *а*, *б*.

Из сопоставления выражений (6.1) – (6.3) и (6.4) – (6.6) следует, что каждому значению порога θ нейрона может быть поставлен в соответствие весовой коэффициент w_0 связи сигнала смещения и наоборот [14-18].

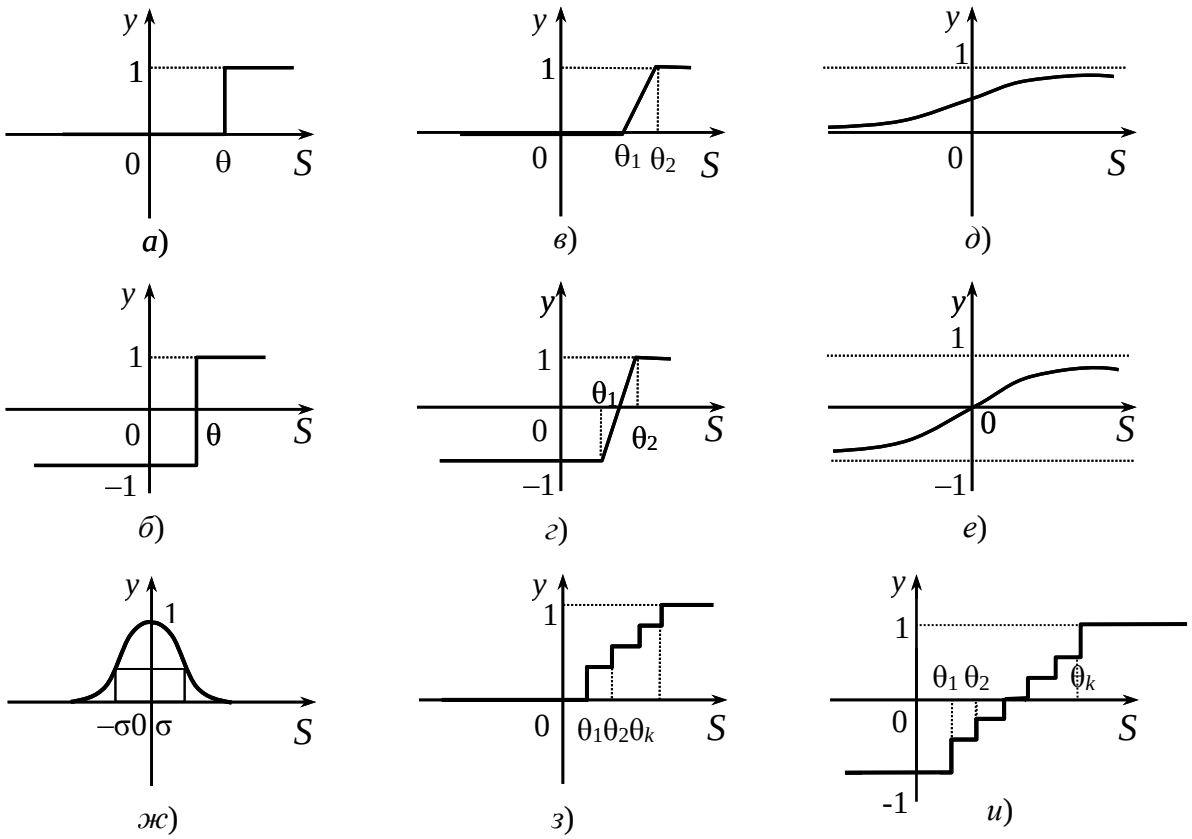


Рис. 6.2. Функции активации нейрона

Реже используются линейные бинарные или биполярные функции активации (рис. 6.2, *в*, *г*):

$$y = \begin{cases} -a \quad \text{при} \quad S < \theta_1, \\ k S + a_0 \quad \text{при} \quad \theta_1 \le S \le \theta_2, \\ 1 \quad \text{при} \quad S > \theta_2, \end{cases} \tag{6.7}$$

где $a = 0$ для бинарных выходных сигналов нейронов и $a = -1$ для биполярных сигналов; k, a_0– постоянные коэффициенты [14].

Кроме приведенных в теории нейронных сетей используются также следующие нелинейные функции активации:

– бинарная сигмоидальная или логическая сигмоидальная (рис. 6.2, *д*):

$$y = \frac{1}{1 + e^{-\tau S}}, \tag{6.8}$$

где τ – постоянный коэффициент;

– биполярная сигмоидальная (рис. 6.2, *е*):

$$y = \frac{2}{1 + e^{-\tau S}} - 1, \tag{6.9}$$

– радиально-симметричная (рис. 6.2, *ж*):

$$y = e^{-\frac{S^2}{\tau^2}}, \tag{6.10}$$

– *К*-значная бинарная (рис. 6.2, *з*):

$$y=\begin{cases}0 \quad \text{при} \quad S<\theta_1,\\ 1/(K-1) \quad \text{при} \ \theta_1 \le S<\theta_2,\\ 2/(K-1) \quad \text{при} \ \theta_2 \le S<\theta_3,\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ \dfrac{K-2}{K-1} \quad \text{при} \quad \theta_{k-1} \le S<\theta_k,\\ 1 \quad \text{при} \quad S \ge \theta_k,\end{cases} \tag{6.11}$$

– *K*-значная биполярная (рис. 6.2, *и*):

$$y=\begin{cases}-1 \quad \text{при} \quad S<\theta_1,\\ -1+2/(K-1) \quad \text{при} \quad \theta_1 \le S<\theta_2,\\ -1+4/(K-1) \quad \text{при} \quad \theta_2 \le S<\theta_3,\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ -1+\dfrac{2(K-2)}{K-1} \quad \text{при} \quad \theta_{k-1} \le S<\theta_k,\\ 1 \quad \text{при} \quad S \ge \theta_k.\end{cases} \tag{6.12}$$

Приведенные модели искусственных нейронов игнорируют многие известные свойства биологических прототипов. Например, они не учитывают временные задержки нейронов, эффекты частотной модуляции, локального возбуждения и связанные с ними явления подпороговой временной и пространственной суммации, когда клетка возбуждается не одновременно пришедшими импульсами, а последовательностями возбуждающих сигналов, поступающих через короткие промежутки времени. Не учитываются также периоды абсолютной рефрактерности, во время которых нервные клетки не могут быть возбуждены, то есть как бы обладают бесконечно большим порогом возбуждения, который затем за несколько миллисекунд после прохождения сигнала снижается до нормального уровня. Этот список отличий, которые многие биологи считают решающими, легко продолжить, однако искусственные нейронные сети все же обнаруживают ряд интересных свойств, характерных для биологических прототипов.

6.4. Нейронная сеть Хебба

Искусственные нейронные сети, предназначенные для решения разнообразных конкретных задач, могут содержать от нескольких нейронов до тысяч и даже миллионов элементов. Однако уже отдельный нейрон (рис. 6.1) с биполярной или бинарной функцией активации может быть использован для решения простых задач распознавания и классификации изображений. Выбор биполярного (1, –1) или бинарного (1, 0) представления сигналов осуществляется исходя из решаемой задачи и во многих случаях он равноценен. Имеется спектр задач, в которых бинарное кодирование сигналов более удобно, однако в общем биполярное представление информации более предпочтительно [13].

Поскольку выходной сигнал у двоичного нейрона (рис. 6.1) принимает только два значения, то нейрон можно использовать для классификации предъявляемых изображений на два класса.

Пусть имеется множество M изображений, для которых известна корректная классификация на два класса

$$X^1 = \{X^{11}, X^{12}, \ldots, X^{1q}\},$$
$$X^2 = \{X^{21}, X^{22}, \ldots, X^{2p}\},$$
$$X^1 \cup X^2 = M, \; X^1 \cap X^2 = \varnothing,$$

и пусть первому классу X^1 соответствует выходной сигнал $y = 1$, а классу X^2 – сигнал $y = -1$. Если, например, предъявлено некоторое изображение $X^\alpha = (X_1^\alpha, \ldots, X_n^\alpha), \; X^\alpha \in M$ и его взвешенная сумма входных сигналов превышает нулевое значение

$$S = \sum_{i=1}^{n} x_i^\alpha w_i + w_0 > 0,$$

то выходной сигнал $y = 1$ и, следовательно, входное изображение X^α

принадлежит классу X^1. Если $S \le 0$, то $y = -1$ и изображение принадлежит второму классу.

Возможно использование отдельного нейрона и для выделения из множества классов

$$M = \left\{X^1 = \left\{X^{11}, X^{12}, ..., X^{1k}\right\}, ..., X^i = \left\{X^{i1}, ..., X^{1q}\right\}, ..., X^p = \left\{X^{p1}, ..., X^{pq}\right\}\right\}$$

изображений единственного класса X^i. В этом случае полагают, что один из двух возможных выходных сигналов нейрона (например, 1) соответствует классу X^i, а второй – всем остальным классам. Поэтому, если входное изображение X^α приводит к появлению сигнала $y = 1$, то $X^\alpha \in X^i$, если $y = -1$ (или $y = 0$, если используется бинарное кодирование), то это означает, что предъявленное изображение не принадлежит выделяемому классу [14].

Система распознавания на основе единственного нейрона делит все пространство возможных решений на две области с помощью гиперплоскости

$$x_1w_1 + x_2w_2 + \ldots + x_nw_n + w_0 = 0.$$

Для двухмерных входных векторов границей между двумя классами изображений является прямая линия: входные вектора, расположенные выше этой прямой, принадлежат к одному классу, а ниже – к другому.

Для адаптации, настройки или обучения весов связей нейрона может использоваться несколько методов. Рассмотрим один из них, получивший название “правило Хебба”. Хебб, исследуя механизмы функционирования центральной нервной системы, предположил, что обучение происходит путем усиления связей между нейронами, активность которых совпадает по времени. Хотя в биологических системах это предположение выполняется далеко не всегда и не исчерпывает всех видов обучения, однако при обучении однослойных нейросетей с биполярными сигналами оно весьма эффективно.

В соответствии с правилом Хебба, если предъявленному биполярному

изображению $X = (x_1, \ldots, x_n)$ соответствует неправильный выходной сигнал y, то веса $w_i \; i = \overline{1,n}$ связей нейрона адаптируются по формуле

$$w_i(t+1) = w_i(t) + x_i y, \quad i = \overline{0,n}, \tag{6.13}$$

где $w_i(t)$, $w_i(t+1)$ соответственно вес i-й связи нейрона до и после адаптации; x_i $(i = \overline{1,n})$ – компоненты входного изображения; $x_0 = 1$ – сигнал смещения; y – выходной сигнал нейрона.

В более полной и строгой форме алгоритм настройки весов связей нейрона с использованием правила Хебба выглядит следующим образом:

Шаг 1. Задается множество $M = \{(X^1, t^1), \ldots, (X^m, t^m)\}$, состоящее из пар (входное изображение $X^k = \{x_1^k, \ldots, x_n^k\}$, необходимый выходной сигнал нейрона t^k), $k = \overline{1,m}$). Инициируются веса связей нейрона:

$$w_i = 0, \quad i = \overline{0,n}.$$

Шаг 2. Для каждой пары (X^k, t^k), $k = \overline{1,m}$, $t^k \in \{1, -1\}$, пока не соблюдаются условия останова, выполняются шаги 3 – 5.

Шаг 3. Инициируется множество входов нейрона:

$$x_0 = 1, \; x_i = x_i^k, \; i = \overline{1,n}.$$

Шаг 4. Инициируется выходной сигнал нейрона: $y = t^k$.

Шаг 5. Корректируются веса связей нейрона по правилу

$$w_i(new) = w_i(old) + x_i y, \quad i = \overline{0,n}.$$

Шаг 6. Проверка условий останова.

Для каждого входного изображения X^k рассчитывается соответствующий ему выходной сигнал y^k:

$$y^k = \begin{cases} 1, & \text{если} \quad S^k > 0, \\ -1, & \text{если} \quad S^k \le 0, \end{cases} \qquad k = \overline{1,m},$$

где

$$S^k = \sum_{i=1}^{n} x_i^k w_i + w_0.$$

Если вектор $(y^1, \ldots, y^m)$ рассчитанных выходных сигналов равен вектору $(t^1, \ldots, t^m)$ заданных сигналов нейрона, т.е. каждому входному изображению соответствует заданный выходной сигнал, то вычисления прекращаются (переход к шагу 7), если же $(y^1, \ldots, y^m) \neq (t^1, \ldots, t^m)$, то переход к шагу 2.

Шаг 7. Останов.

Пример 6.1. Пусть требуется обучить биполярный нейрон распознаванию изображений X^1 и X^2, приведенных на рис. 6.3.

X^1

1	2	3
4	5	6
7	8	9

X^2

1	2	3
4	5	6
7	8	9

Рис. 6.3. Входные изображения

При этом нужно чтобы изображению X^1 соответствовал выходной сигнал нейрона "+1", а изображению X^2 – сигнал "–1".

Применение алгоритма Хебба дает следующие результаты.

Шаг 1. Задается множество

$$M = \{(X^1 = (1, -1, 1, 1, 1, 1, -1, -1, 1), 1),$$
$$(X^2 = (1, 1, 1, 1, -1, 1, 1, -1, 1), -1)\};$$

инициируются веса связей нейрона: $w_i = 0,\ i = \overline{0, 9}.$

Шаг 2. Для каждой из двух пар (X^1, 1), (X^2, –1), выполняются шаги 3-5.

Шаг 3. Инициируется множество входов нейрона для изображения первой пары: $x_0 = 1,\ x_i = x_i^1,\ i = \overline{0, 9}.$

Шаг 4. Инициируется выходной сигнал нейрона для изображения первой пары: $y = t^1 = 1.$

Шаг 5. Корректируются веса связей нейрона по правилу Хебба $w_i = w_i + x_i^1 y \quad (i = \overline{0, n})$:

$$w_0 = w_0 + x_0 y = 0 + 1 \cdot 1 = 1;$$
$$w_1 = w_1 + x_1^1 y = 0 + 1 \cdot 1 = 1;$$
$$w_1 = w_3 = w_4 = w_5 = w_6 = w_9 = 1;$$
$$w_2 = w_2 + x_2^1 y = 0 + (-1) \cdot 1 = -1;$$
$$w_2 = w_7 = w_8 = -1.$$

Шаг 3. Инициируется множество входов нейрона для изображения X^2 второй пары: $x_0 = 1,\ x_i = x_i^2,\ i = \overline{0, 9}.$

Шаг 4. Инициируется выходной сигнал нейрона для изображения второй пары (X^2, t^2):

$$y = t^2 = -1.$$

Шаг 5. Корректируются веса связей нейрона:

$$w_0 = w_0 + x_0 y = 1 + 1 \cdot (-1) = 0;$$

$$w_1 = w_1 + x_1^2 y = 1 + 1 \cdot (-1) = 0;$$

$$w_1 = w_3 = w_4 = w_6 = w_9 = 0;$$

$$w_2 = w_2 + x_2^2 y = -1 + 1 \cdot (-1) = -2;$$

$$w_2 = w_7 = -2;$$

$$w_5 = w_5 + x_5^2 y = 1 + (-1) \cdot (-1) = 2;$$

$$w_8 = w_8 + x_8^2 y = -1 + (-1) \cdot (-1) = 0.$$

Шаг 6. Проверяются условия останова.

Рассчитываются входные и выходной сигналы нейрона при предъявлении изображения X^1:

$$S^1 = \sum_{i=1}^{9} x_i^1 w_i + w_0 = 1 \cdot 0 + (-1) \cdot (-2) + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 2 + 1 \cdot 0 + (-1) \cdot (-2) +$$

$$+ (-1) \cdot 0 + 1 \cdot 0 + 0 = 6,$$

$$y^1 = 1, \text{ так как } S^1 > 0.$$

Рассчитываются входной и выходной сигналы нейрона при предъявлении изображения X^2

$$S^2 = \sum_{i=1}^{9} x_i^2 w_i + w_0 = 1 \cdot 0 + 1 \cdot (-2) + 1 \cdot 0 + 1 \cdot 0 + (-1) \cdot 2 + 1 \cdot 0 + 1 \cdot (-2) +$$

$$+ (-1) \cdot 0 + 1 \cdot 0 + 0 = -6,$$

$$y^2 = -1, \text{ так как } S^2 < 0.$$

Поскольку вектор (y^1, y^2) = (1, –1) равен вектору (t^1, t^2), то вычисления прекращаются, так как цель достигнута – нейрон правильно распознает заданные изображения [14].

Шаг 7. Останов.

Основная идея правила (6.13) – усиливать связи, которые соединяют нейроны с одинаковой по времени активностью, и ослаблять связи, соединяющие элементы с различной активностью, может быть использована и при настройке нейросетей с бинарными элементами. Правило Хебба (6.13) для однослойных бинарных сетей можно записать в виде:

$$w_i\,(t+1) = w_i\,(t) + \Delta w_i, \tag{6.14}$$

где

$$\Delta w_i = \begin{cases} 1, & \text{если} \quad x_i\, y = 1, \\ 0, & \text{если} \quad x_i = 0, \\ -1, & \text{если} \quad x_i \neq 0 \quad \text{и} \quad y = 0. \end{cases} \tag{6.15}$$

Пример 6.2. Пусть требуется обучить бинарный нейрон распознаванию изображений X^1и X^2 примера 6.1. При этом изображению X^1 пусть соответствует выходной сигнал нейрона “+1”, а изображению X^2 – “0”. Применение правила Хебба в этом случае дает следующие результаты:

Шаг 1. Задается множество

$$M = \{(X^1 = (1, 0, 1, 1, 1, 1, 0, 0, 1), 1),$$
$$(X^2 = (\,1, 1, 1, 1, 0, 1, 1, 0, 1\,), 0)\},$$

и инициируются веса связей нейрона $w_i = 0,\ i = \overline{0,9}.$

Шаг 2. Для пар $(X^1,1)$, $(X^2, 0)$, выполняются шаги 3 – 5.

Шаг 3. Инициируется множество входов нейрона элементами изображения X^1:

$$x_0 = 1,\ x_i = x_i^1,\ i = \overline{0,9}.$$

Шаг 4. Инициируется выходной сигнал нейрона для изображения X^1:

$$y = t^1 = 1.$$

Шаг 5. Корректируются веса связей нейрона с помощью соотношений (6.14), (6.15):

$$w_0 = w_0 + \Delta w_0 = 0 + 1 = 1;$$

$$w_1 = w_1 + \Delta w_1 = 0 + 1 = 1;$$

$$w_1 = w_3 = w_4 = w_5 = w_6 = w_9 = 1;$$

$$w_2 = w_2 + \Delta w_2 = 0 + 0 = 0;$$

$$w_2 = w_7 = w_8 = 0.$$

Шаг 3. Инициируется множество входов нейрона элементами изображения X^2:

$$x_0 = 1,\ x_i = x_i^2\ i = \overline{0, 9}.$$

Шаг 4. Инициируется выходной сигнал нейрона для изображения X^2:

$$y = t^2 = 0.$$

Шаг 5. Корректируются веса связей нейрона с помощью соотношений (6.14), (6.15):

$$w_0 = w_0 + \Delta w_0 = 1 + (-1) = 0;$$

$$w_1 = w_1 + \Delta w_1 = 1 + (-1) = 0;$$

$$w_1 = w_3 = w_4 = w_6 = w_9 = 0;$$

$$w_2 = w_2 + \Delta w_2 = 0 + (-1) = -1;$$

$$w_5 = w_5 + \Delta w_5 = 1 + 0 = 1;$$

$$w_7 = w_2 = -1;$$

$$w_8 = w_8 + \Delta w_8 = 0 + 0 = 0.$$

Шаг 6. Проверка условий останова.

Рассчитываются входные и выходные сигналы нейрона при предъявлении изображений X^1, X^2:

$$S^1 = \sum_{i=1}^{9} x_i^1 w_i + w_0 = 1\cdot 0 + 0\cdot(-1) + 1\cdot 0 + 1\cdot 0 + 1\cdot 1 + 1\cdot 0 + 0\cdot(-1) + $$
$$+ 0\cdot 0 + 1\cdot 0 + 0 = 1;$$

$$y^1 = 1, \text{ так как } S^1 > 0;$$

$$S^2 = \sum_{i=1}^{9} x_i^2 w_i + w_0 = 1\cdot 0 + 1\cdot(-1) + 1\cdot 0 + 1\cdot 0 + 0\cdot 1 + 1\cdot 0 + 1\cdot(-1) + $$
$$+ 0\cdot 0 + 1\cdot 0 = -2;$$

$$y^2 = -1, \text{ так как } S^2 < 0.$$

Поскольку вектор $(y^1, y^2) = (1, 0)$ равен заданному вектору $(t^1, t^2) = (1, 0)$, то цель достигнута и вычисления прекращаются.

Шаг 7. Останов.

Использование группы из m биполярных или бинарных нейронов $A_1, \ldots, A_m$ (рис. 6.4) позволяет существенно расширить возможности нейронной сети и распознавать до 2^m различных изображений. Правда, применение этой сети для распознавания 2^m (или близких к 2^m чисел) различных изображений может приводить к неразрешимым проблемам адаптации весов связей нейросети. Поэтому часто рекомендуют использовать данную архитектуру для распознавания только m различных изображений, задавая каждому из них единичный выход только на выходе одного A-элемента (выходы остальных при этом должны принимать значение “–1” для биполярных нейронов или “0” – для бинарных).

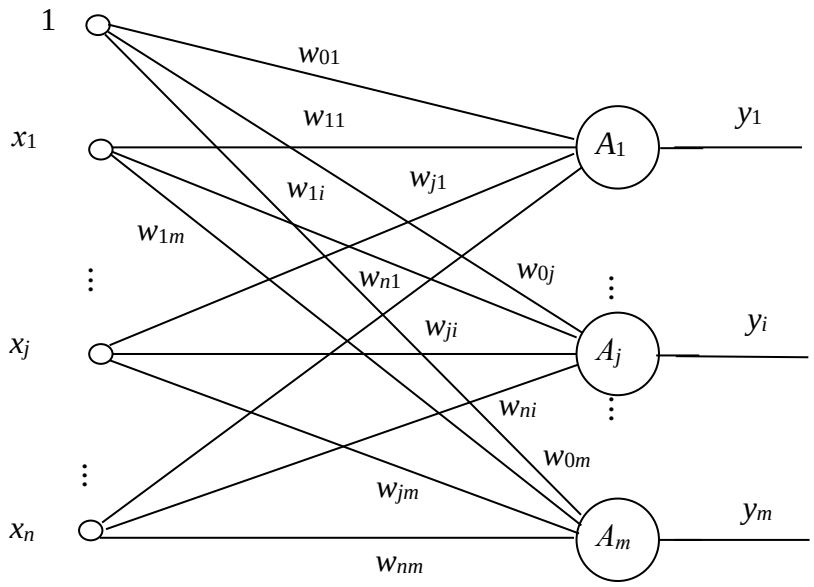


Рис. 6.4. Нейронная сеть из m элементов

Однослойная нейронная сеть с двоичными нейронами, приведенная на рис. 6.4, может быть обучена с помощью алгоритма на основе правила Хебба. В этом случае она называется сетью Хебба. Использование других алгоритмов обучения этой же сети приводит и к изменению названия нейронной сети. Использование в названии сетей их алгоритмов обучения характерно для теории нейронных сетей. Для биполярного представления сигналов возможно обучение нейросети с помощью следующего алгоритма:

Шаг 1. Задается множество $M = \{(X^1, t^1), \ldots, (X^m, t^m)\}$, состоящее из пар (входное изображение $X^k = \left(x_1^k, \ldots, x_n^k\right)$, необходимый выходной сигнал нейрона $t^k, k = \overline{1, m}$). Инициируются веса связей нейрона:

$$w_{ji} = 0,\ j = \overline{1, n},\ \ i = \overline{1, m}.$$

Шаг 2. Каждая пара(X^k, t^k), проверяется на правильность реакции нейронной сети на входное изображение. Если полученный выходной вектор

сети (y_1^k, ..., y_m^k), отличается от заданного $t^1 = (t_1^k, ..., t_m^k)$, то выполняют шаги 3 – 5.

Шаг 3. Инициируется множество входов нейронов: $x_0 = 1$, $x_j = x_j^k$, $j = \overline{1,n}$.

Шаг 4. Инициируются выходные сигналы нейронов: $y_i = t_i^k$, $i = \overline{0,m}$.

Шаг 5. Корректируются веса связей нейронов по правилу:

$$w_{ji}(new) = w_{ji}(old) + x_j y_i, \quad j = \overline{0,n}, \quad i = \overline{0,m}.$$

Шаг 6. Проверяются условия останова, т.е. правильности функционирования сети при предъявлении каждого входного изображения. Если условия не выполняются, то переход к шагу 2 алгоритма, иначе – прекращение вычислений (переход к шагу 7).

Шаг 7. Останов.

6.5. Перцептроны – класс моделей мозга

Перцептроны, или персептроны (от *perceptio* – восприятие) были первыми искусственными нейронными сетями, появившимися в результате многолетних исследований мозга животных и человека. Автор первого перцептрона – американский ученый Френк Розенблатт, впервые опубликовавший свои исследования в этой области в 1957 году. По мнению Ф. Розенблатта, перцептроны, прежде всего, являются классом моделей мозга, объясняющих некоторые его характерные функции. В частности, перцептроны, пусть и в самой элементарной форме, объясняют некоторые проблемы организации памяти биологических систем, демонстрируют механизм приобретения знаний “познающих (*cognitive*) систем” об окружающем их мире и показывают, что эти знания зависят как от когнитивной системы, так и от окружающей среды [14-18]. По Розенблатту, для различных видов животных простейшее представление об анатомической структуре нервной системы может быть получено с помощью схемы, показанной на рис. 6.5.

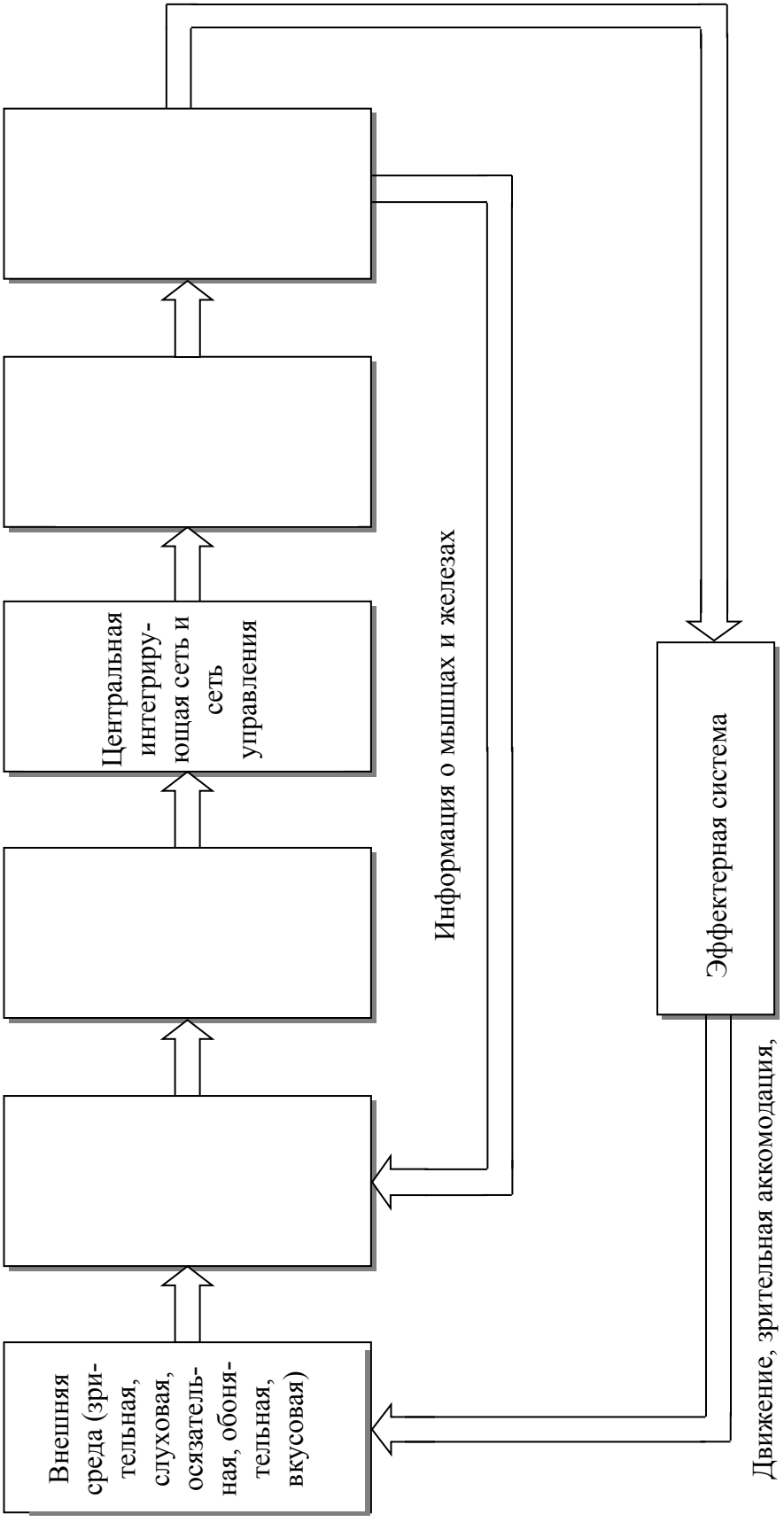


Рис. 6.5. Структура нервной системы

В своих первых работах Розенблатт рассматривал модель только зрительной системы. В наиболее простом виде эта модель включает в себя три последовательно соединенных множества нейронов: чувствительных (*S*-элементов), ассоциирующих (*A*-элементов) и реагирующих (*R*-элементов). *S*-элементам в нервной системе животного или человека соответствуют сенсорные или рецепторные нейроны, генерирующие сигналы на поступающие внешние раздражения (изображения) и передающие их *A*-нейронам.

A-элементы аналогичны в нервной системе живого организма нейронам, образующим локальный специализированный зрительный центр в коре головного мозга и связывающим рецепторные нейроны с моторными. *R*-элементам в нервной системе соответствуют эффекторные (моторные) нейроны, упорядоченные в ограниченные топологические структуры и передающие сигналы управления центральной нервной системы к мышцам и железам организма.

Определение 6.1. *S*-элемент называется простым, если он выдает единичный выходной сигнал при входном сигнале, превышающем некоторый заданный порог θ, и нулевой сигнал – в противном случае.

Определение 6.2. Простым ассоциативным элементом называется *A*-элемент, который выдает единичный выходной сигнал, если алгебраическая сумма его входных сигналов превышает некоторый заданный порог $\theta > 0$, иначе – выходной сигнал ассоциативного нейрона равен нулю.

Определение 6.3. Простым биполярным (бинарным) реагирующим элементом называется *R*-элемент, выдающий единичный выходной сигнал, если алгебраическая сумма его входных сигналов больше или равна пороговому значению, и отрицательный единичный (нулевой) сигнал, если сумма его входных сигналов меньше заданного порога.

Чувствительные *S*-элементы живого организма (рис. 6.6) возбуждаются от воздействия энергии света, если величины их входных сигналов превышают некоторый порог θ_i.

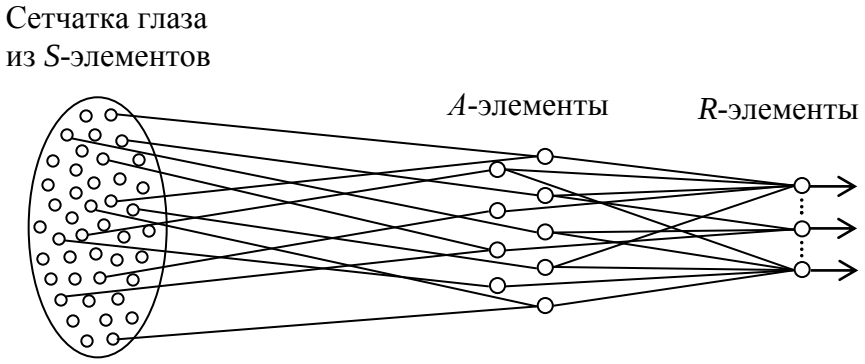


Рис. 6.6. Структура модели зрительной системы

Рецепторные нейроны случайным образом связаны с *A*-элементами, выходные сигналы которых отличны от нуля только в том случае, когда возбуждено достаточно большое число сенсорных нейронов, воздействующих на входы одного ассоциирующего элемента.

Простой *A*-элемент, аналогично простому *S*-элементу, является активным и выдает единичный выходной сигнал, если алгебраическая сумма сигналов на его входе превышает заданную пороговую величину, в противном случае нейрон находится в невозбужденном состоянии.

Коэффициенты (веса) связей между *S*- и *A*-элементами постоянны.

Комбинация выходов всех *A*-элементов представляет собой реакцию двух первых слоев перцептрона на предъявленное входное изображение, которая с помощью выходного слоя нейронов преобразуется в необходимую комбинацию выходных сигналов системы. Часто требуют, чтобы каждому классу входных изображений соответствовал только один определенный активный *R*-нейрон. Необходимых комбинаций выходных сигналов на каждый класс изображений добиваются на этапе обучения или адаптации перцептрона за счет изменения переменных весов связей между *A*- и *R*-элементами.

Разделение множества G изображений на два класса G_1 и G_2 можно выполнить с помощью одного выходного элемента. В этом случае изображениям первого класса может соответствовать положительный выходной сигнал (+1) *R*-элемента, а изображениям второго класса – отрицательный (–1). На примере

простейшего (элементарного) перцептрона рассмотрим различные способы обучения этих нейросетей, впервые предложенные и исследованные Розенблаттом.

Определение 6.4. Простым перцептроном называется нейронная сеть, состоящая из *S*-, *A*- и *R*-элементов и удовлетворяющая следующим пяти условиям:

1. В сети имеется только один *R*-нейрон, который соединен связями с переменными весами со всеми *A*-нейронами.

2. В сети имеются только последовательные связи от *S*- к *A*-элементам и от *A*-элементов к *R*-элементу.

3. Веса связей между *S*- и *A*-элементами являются фиксированными.

4. Время передачи сигналов каждой связью равно нулю (либо фиксированной постоянной величине).

5. Выходные сигналы всех нейронов сети формируются в виде

$$U_{\text{вых}} = f\left(\sum_i U_{\text{вх}.i}(t)\right),$$

где $\sum_i U_{\text{вх}.i}(t)$ – алгебраическая сумма сигналов, поступающих одновременно на вход нейрона.

Определение 6.5. Простой перцептрон с простыми *A*- и *R*-элементами и передающими функциями связей вида:

$$C_{ij}(t) = w_{ij}(t) U_{\text{вых}.i}(t - \tau_{ij}),$$

где $w_{ij}(t)$ – вес связи между *i*-м и *j*-м нейронами в момент времени *t*; $U_{\text{вых}.i}(t - \tau_{ij})$ – выходной сигнал *i*-го нейрона в момент времени $(t - \tau_{ij})$; τ_{ij} – время передачи сигнала $U_{\text{вых}.i}(t - \tau_{ij})$ с выхода *i*-го нейрона на вход *j*-го элемента, называется элементарным перцептроном.

Элементарный перцептрон обучается или настраивается на распознава-

ние двух классов изображений G_1, G_2 путем предъявления ему некоторых последовательностей изображений из этих классов. Учитель (человек или компьютер), наблюдающий реакцию перцептрона на каждое входное изображение, при наличии ошибочных решений сети должен корректировать веса связей между *R*- и *A*-элементами в соответствии с некоторой системой правил.

Определение 6.6. Матрицей взаимодействия перцептрона называется матрица, элементами которой являются веса связей w_{ij} для всех пар нейронов U_i, U_j сети.

Если связь между нейронами U_i, U_j отсутствует (например, в простом перцептроне нет связей между *R*- и *S*-нейронами), то принимают $w_{ij} = 0$.

Матрица взаимодействия фактически отображает состояние памяти перцептрона. Множество всех возможных состояний памяти сети образует фазовое пространство сети, которое может быть представлено в виде области в *n*-мерном евклидовом пространстве, каждая координатная ось которого соответствует одной связи сети.

6.6. Обучение перцептронов с помощью α- и γ-систем подкрепления

Определение 6.7. Системой подкрепления нейронной сети называется любой набор правил, с помощью которых можно изменять во времени состояние памяти сети (или матрицу взаимодействия).

Определение 6.8. Положительным (отрицательным) подкреплением называется такой процесс коррекции весов связей, при котором вес связи $w_{ij}(t)$, начинающейся на выходе активного *i*-го элемента и оканчивающейся на входе *j*-го элемента, изменяется на величину $\Delta w_{ij}(t)$, знак которой совпадает со знаком выходного сигнала *j*-го нейрона (знак которой противоположен знаку выходного сигнала *j*-го нейрона).

Существует множество различных систем подкрепления. Остановимся только на системе подкрепления с коррекцией ошибок, которая является основ-

ной в настоящее время. В системе подкрепления с коррекцией ошибок прежде всего необходимо определить, является ли реакция перцептрона правильной. До тех пор пока выходной сигнал *R*-элемента принимает желаемое значение, величина сигнала подкрепления η равна нулю. При появлении неправильной реакции перцептрона используется подкрепление, величина и знак которого в общем случае определяется монотонно возрастающей функцией *f*:

$$\eta = f(R^* - R), \tag{6.16}$$

где R^* – желаемая реакция; R – полученная реакция; $f(0) = 0$.

Таким образом, при появлении ошибки для коррекции весов связей используется сигнал, знак которого противоположен знаку выходного сигнала *R*-элемента. В связи с этим рассмотренный метод коррекции весов получил название системы с отрицательным подкреплением.

Конкретным примером системы подкрепления с коррекцией ошибок является альфа-система подкрепления. В этой системе при наличии ошибок веса всех активных связей, которые оканчиваются на *R*-элементе, изменяют на одинаковую величину η, а веса всех неактивных связей оставляют без изменений. Перцептроны, в которых применяется альфа-система подкрепления, называются альфа-перцептронами.

При использовании альфа-системы подкрепления сумма весов всех связей между *R*- и *A*-нейронами может возрастать (или убывать) от шага к шагу, что должно приводить к нежелательным ситуациям, когда многие связи имеют максимальные (или минимальные) веса и не могут использоваться в дальнейшем процессе обучения нейронной сети. Для устранения этого недостатка α-системы подкрепления была предложена гамма-система подкрепления, которая обладает свойством консервативности относительно суммы Σ_1 весов всех связей между нейронами, тоесть сумма Σ_1 остается постоянной в процессе обучения перцептрона. Это достигается за счет того, что при наличии ошибочной реакции перцептрона сначала веса всех активных связей изменяются на одинако-

вое значение η, а вслед за этим из весов всех активных и пассивных связей вычитается величина, равная отношению суммы изменения весов всех активных связей к числу всех связей. Изменение весов отдельных связей при этом определяется соотношением

$$\Delta w_{ij} = \begin{cases} \eta - \dfrac{N_{ак}\eta}{N}, & \text{если } i\text{-й } A\text{-нейрон возбужден,} \\ \\ -\dfrac{N_{ак}\eta}{N}, & \text{если } i\text{-й } A\text{-нейрон заторможен,} \end{cases} \tag{6.17}$$

где Δw_{ij} – в общем случае приращение веса связи между i-м A-нейроном и j-м R-нейроном, для элементарного перцептрона $j = \text{const} = 1$; η – величина сигнала подкрепления; $N_{ак}$ – число активных связей; N – число связей, оканчивающихся на входе j-го элемента.

При такой системе коррекции весов связей выполняется равенство:

$$\eta N_{ак} - \frac{N_{ак}\eta}{N} \cdot N = 0,$$

из которого и следует консервативность гамма-системы подкрепления относительно суммы весов всех обучаемых связей.

Отметим, что соотношение (6.17) в неявной форме предполагает, что корректируемые веса w_{ij} связей достаточно далеки от своих граничных значений $w_{ij\,\min} = 0$ и $w_{ij\,\max} = 1$, то есть

$$w_{ij\,\min} \le w_{ij} + \Delta w_{ij} \le w_{ij\,\max}\ . \tag{6.18}$$

Если неравенства (6.18) нарушаются, а требование консервативности относительно суммы Σ_1 весов связей остается неизменным, то соотношение (6.17)

необходимо уточнить. Пусть, например, среди активных связей $N_{a\,\text{гр}}$ связей имеют граничные значения весов $w_{ij\max}$ или $w_{ij\min}$ и для них выполняются условия

$$w_{ij\max} + \Delta w_{ij} > w_{ij\max} \quad \text{или} \quad w_{ij\min} + \Delta w_{ij} < w_{ij\min}\,. \qquad (6.19)$$

Пусть также $N_{a\,\text{бгр}}$ активных связей имеют веса, близкие к граничным, для которых справедливы неравенства

$$w_{ij} + \Delta w_{ij} > w_{ij\max} \quad \text{или} \quad w_{ij} + \Delta w_{ij} < w_{ij\min}\,. \qquad (6.20)$$

В этом случае общая сумма S_a первоначальных изменений весов активных связей будет равна:

$$S_a = (N_{ак} - N_{a\,гр} - N_{a\,бгр})\eta + \operatorname{sign}(\eta) \sum_{k=1}^{N_{a\,бгр}} \left| w_{kj}^{гр.} - w_{kj} \right|, \quad \left| w_{kj}^{гр.} - w_{kj} \right| < \left| \Delta w_{kj} \right|, \qquad (6.21)$$

где $w_{kj}^{гр.}$ – граничное значение веса связи между k-м и j-м нейронами, $w_{kj}^{гр.} \in \{0,1\}$; Δw_{kj} – приращения веса связи, определяемое по соотношению (6.17) без учета наличия множества $\{w_{ij\min},\ w_{ij\max}\} \equiv \{0,1\}$; $\operatorname{sign}(\cdot)$ – знаковая функция.

Если предположить, что для всех пассивных связей выполняются соотношения (6.18), тогда из весов пассивных связей и весов активных связей, для которых не выполняется соотношение (6.19) или (6.20), вычитается величина

$$S_a = \left| (N_{ак} - N_{a\,гр} - N_{a\,бгр}) \right| \eta\,.$$

С учетом этих замечаний соотношение (6.17) принимает вид

$$\Delta w_{ij} = \begin{cases} \eta_1 = \eta - \dfrac{S_a}{(N_{ак} - N_{а\,гр} - N_{а\,бгр})}, \dfrac{\text{если } i\text{-й } A\text{-нейрон возбужден и не}}{\text{выполняются условия (6.19) или (6.20)}}; \\ 0, \text{ если } i\text{-й } A\text{-нейрон возбужден и выполняется одно из условий (6.19)}; \\ 1 - w_{ij}, \text{ если } w_{ij} + \eta_1 > 1 \text{ и } i\text{-й } A\text{-нейрон возбужден}; \\ 0 - w_{ij}, \text{ если } w_{ij} - \eta_1 < 0 \text{ и } i\text{-й } A\text{-нейрон возбужден}; \\ -S_a/(N_{ак} - N_{а\,гр} - N_{а\,бгр}), \text{ если } i\text{-й } A\text{-нейрон заторможен}. \end{cases}$$

Примером еще одного общего способа обучения перцептронов является метод коррекции ошибок случайными возмущениями. Он предусматривает, как и альфа-система подкрепления, при появлении ошибок – коррекцию весов активных связей, но знак и величина коррекции для каждой связи выбирается случайно в соответствии с некоторым заданным распределением вероятностей.

Пример 6.3. Выполним обучение элементарного перцептрона с бинарными *S*- и *A*-нейронами и биполярным *R*-нейроном (рис. 6.7) распознаванию изображений букв Н и П (рис. 6.8, *а*, *б*) на рецепторном поле из девяти элементов (рис. 6.8, *в*) [14]. При этом потребуем, чтобы при предъявлении изображения буквы Н на выходе *R*-элемента был сигнал “–1”, при появлении второго изображения – сигнал “+1”.

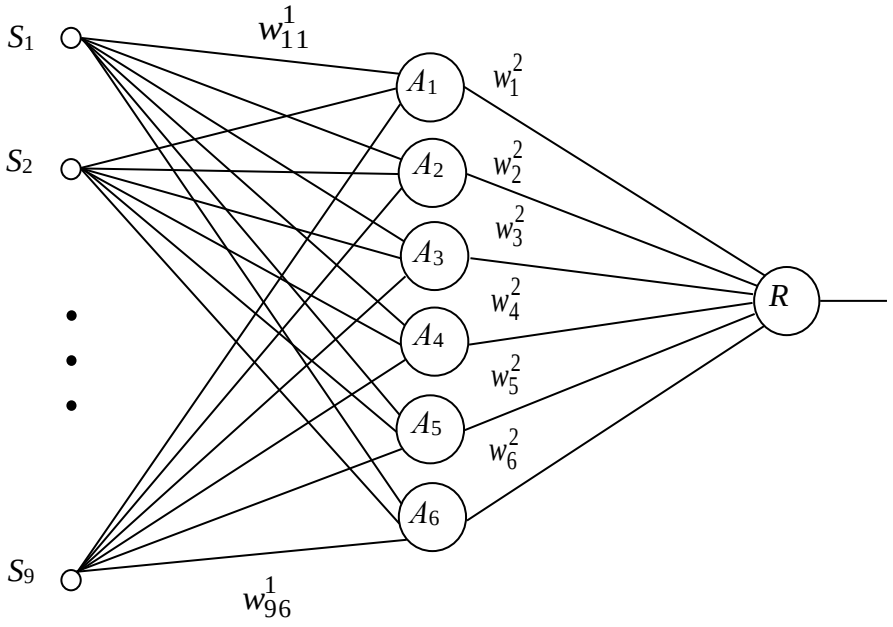
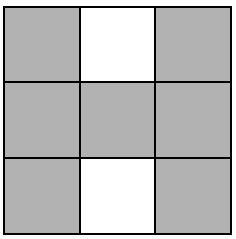
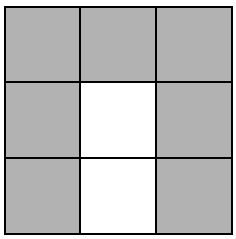


Рис. 6.7. Элементарный перцептрон



а



б

1	2	3
4	5	6
7	8	9

в

Рис. 6.8. Изображения букв Н и П

Зададим в табл. 6.1 и 6.2 веса связей w_{ij}^1 ($i=\overline{1,9}$, $j=\overline{1,6}$), w_k^2 ($k=\overline{1,6}$) соответственно между бинарными *S*- и *A*-нейронами и между *A*-нейронами и биполярным нейроном *R* с помощью генератора случайных чисел, генерирующего их из конечного множества {0,1; 0,2; …; 0,9}.

Таблица 6.1 – Веса w_{ij}^1 связей перцептрона между *S*- и *A*-элементами

w_{ij}^1	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
A_1	0,3	0,2	0,1	0,6	0,5	0,4	0,9	0,6	0,7
A_2	0,2	0,1	0,3	0,4	0,5	0,6	0,7	0,1	0,9
A_3	0,3	0,5	0,1	0,6	0,5	0,4	0,9	0,4	0,7
A_4	0,4	0,3	0,2	0,1	0,8	0,7	0,6	0,6	0,9
A_5	0,5	0,3	0,3	0,6	0,1	0,2	0,9	0,2	0,7
A_6	0,6	0,5	0,4	0,1	0,2	0,3	0,8	0,5	0,8

Таблица 6.2 – Веса w_k^2 связей перцептрона между *R*- и *A*-элементами

w_k^2	A_1	A_2	A_3	A_4	A_5	A_6
R	0,2	0,8	0,6	0,9	0,8	0,1

Подадим на вход перцептрона изображение буквы Н (рис 6.8, *а*). Это изображение возбуждает все *S*-нейроны, кроме второго и восьмого. Единичные сигналы с выходов возбужденных бинарных *S*-нейронов через связи, весовые коэффициенты которых заданы табл. 6.1, поступают на входы *A*-нейронов. Суммарный входной сигнал на входе *i*-го *A*-элемента определяется соотношением

$$U_{вх.\,Ai} = \sum_{j=1}^{9} U_{вых.\,Sj} w_{ji}^{1}, \quad j = \overline{1,6}, \tag{6.22}$$

где $U_{вх.Ai}$ – сигнал на входе i-го A-нейрона; $U_{вых.Sj}$ – сигнал на выходе j-го S-нейрона; w_{ji} – вес связи между j-м S-нейроном и i-м A-элементом.

Для первого A-нейрона имеем

$$U_{вх.A1} = \sum_{j=1}^{9} U_{вых.Sj} w_{j1}^{1} = 1 \cdot 0{,}3 + 0 \cdot 0{,}2 + 1 \cdot 0{,}1 + 1 \cdot 0{,}6 + 1 \cdot 0{,}5 + 1 \cdot 0{,}4 + \\ + 1 \cdot 0{,}9 + 0 \cdot 0{,}6 + 1 \cdot 0{,}7 = 3{,}5.$$

Аналогично вычисляются сигналы на входах остальных A-элементов. Результаты этих вычислений приведены во второй строке табл. 6.3. В третьей строке этой таблицы – результаты расчетов сигналов на входах A-элементов при предъявлении перцептрону изображения буквы П.

Таблица 6.3 – Величины сигналов на входах A-элементов

Изображение	Сигналы на входах A-элементов					
	$U_{вх.A1}$	$U_{вх.A2}$	$U_{вх.A3}$	$U_{вх.A41}$	$U_{вх.A5}$	$U_{вх.A6}$
Буква Н	3,5	3,6	3,5	3,7	3,3	3,2
Буква П	3,2	3,2	3,4	3,2	3,6	3,5

Для упрощения расчетов положим, что пороги θ_i, $i = \overline{1,6}$ всех A-нейронов одинаковы

$$\theta_1 = \theta_2 = \ldots = \theta_6 = \theta.$$

Если величина порога θ выбрана меньше 3,2, то при предъявлении любого изображения будут возбуждены все A-нейроны, а если выбрать $\theta > 3{,}7$, то на выходах всех нейронов будут нулевые сигналы. В обоих этих случаях перцептрон не может выполнять распознавание предъявляемых изображений.

Очевидно, что для обеспечения работоспособности нейронной сети порог θ необходимо выбрать между 3,2 и 3,7 и таким образом, чтобы при предъявлении разных изображений возбуждались различные множества M_1, M_2 A-элементов, причем желательно, чтобы эти множества не пересекались, т.е.

$$M_1 \bigcap M_2 = 0. \tag{6.23}$$

Пусть выходной сигнал A-элементов определяется соотношением

$$U_{вых.A} = \begin{cases} 1, \text{если } U_{вх.A} \geq \theta, \\ 0, \text{если } U_{вх.A} < \theta, \end{cases}$$

тогда условие (6.23) выполняется при $\theta = 3{,}5$ и при предъявлении изображения буквы Н будут возбуждены элементы A_1, A_2, A_3 и A_4, а при предъявлении буквы П – нейроны A_5 и A_6. Рассчитаем с учетом данных табл. 6.3 сигналы $U_{вхR\text{Н}}$, $U_{вхR\text{П}}$ на входе R-нейрона при предъявлении изображений букв Н и П:

$$U_{вх.R\text{Н}} = \sum_{i=1}^{6} U_{вых.Ai} w_i^2 = 1 \cdot 0{,}2 + 1 \cdot 0{,}8 + 1 \cdot 0{,}6 + 1 \cdot 0{,}9 + 0 \cdot 0{,}8 + 0 \cdot 0{,}1 = 2{,}5,$$

$$U_{вх.R\text{П}} = \sum_{i=1}^{6} U_{вых.Ai} w_i^2 = 0 \cdot 0{,}2 + 0 \cdot 0{,}8 + 0 \cdot 0{,}6 + 0 \cdot 0{,}9 + 1 \cdot 0{,}8 + 1 \cdot 0{,}1 = 0{,}9.$$

При величине порога R-элемента $\theta_R = 1{,}7$ и предъявлении изображения буквы Н на выходе перцептрона будет сигнал “+1”, а при предъявлении второго изображения – сигнал “–1”, что не соответствует исходным требованиям к выходным сигналам нейронной сети.

Используем для настройки перцептрона α-систему подкрепления при величине сигнала подкрепления η, равном 0,1 и при предъявлении последовательности изображений Н, П, Н, П, … в моменты времени t_1, t_2, t_3, … .

Процесс адаптации весов связей между R- и A-нейронами иллюстрируется в табл. 6.4.

Таблица 6.4 – Адаптация весов связей перцептрона с помощью α-системы подкрепления

Весовые коэффициенты и входные сигналы	Моменты времени									
	t_0	$t_1^{\text{Н}}$	$t_2^{\text{П}}$	$t_3^{\text{Н}}$	$t_4^{\text{П}}$	$t_5^{\text{Н}}$	$t_6^{\text{П}}$	$t_7^{\text{П}}$	$t_8^{\text{П}}$	$t_9^{\text{П}}$
w_1^2	0,2	0,1	0	0	0	0	0	0	0	0
w_2^2	0,8	0,7	0	0,6	0	0,5	0	0	0	0
w_3^2	0,6	0,5	0	0,4	0	0,3	0	0	0	0
w_4^2	0,9	0,8	0	0,7	0	0,6	0	0	0	0
w_5^2	0,8	0	0,9	0	1	0	1	1	1	1
w_6^2	0,1	0	0,2	0	0,3	0	0,4	0,5	0,6	0,7
$U_{вхR\text{Н}}$	2,5	2,1	–	1,7	–	1,4	–	–	–	–
$U_{вхR\text{П}}$	0,9	–	1,1	–	1,3	–	1,4	1,5	1,6	1,7

Во втором столбце таблицы при $t = t_0$ приведены значения исходных весов связей и величины сигналов $U_{вх.R\text{Н}}$, $U_{вх.R\text{П}}$ на входе *R*-элемента при предъявлении соответственно изображений букв Н и П. При первом предъявлении изображения буквы Н в момент времени t_1 (обозначено $t_1^{\text{Н}}$) в силу наличия ошибочного сигнала на выходе перцептрона корректируются веса активных связей $w_1^2, \ldots, w_4^2$ на величину $\eta = 0{,}1$. Эта коррекция уменьшает суммарный входной сигнал $U_{вх.R\text{Н}}$ до величины 2,1.

Пусть функционирование *R*-элемента описывается соотношением:

$$U_{вых.R} = \begin{cases} +1, \text{ если } U_{вх.R} \geq \theta_R, \\ -1, \text{ если } U_{вх.R} < \theta_R, \end{cases}$$

где θ_R – порог *R*-элемента, тогда для достижения правильной реакции *R*-элемента на изображение буквы Н необходимы две повторные коррекции весов связей $w_1^2, \ldots, w_4^2$.

Результаты этих коррекций приведены в табл. 6.4 соответственно в пятом и седьмом столбцах при $t = t_3^{\text{Н}}$ и $t = t_5^{\text{Н}}$. После момента времени $t = t_5^{\text{Н}}$, так как выполняется соотношение (6.23), из входной последовательности могут быть исключены изображения буквы Н и предъявляться только изображения буквы П. Результаты коррекции весов связей w_5^2, w_6^2, определяющих сигнал на входе *R*-элемента при предъявлении изображения буквы П, приведены в последней строке таблицы. Поскольку в рассматриваемом примере коррекция входного сигнала *R*-элемента при предъявлении изображения буквы П осуществляется только с помощью весов двух связей, причем, после второй коррекции вес связи w_5^2 принимает максимальное значение и в дальнейшем возрастать не может, то процесс обучения нейронной сети правильной реакции на второе изображение более длительный и заканчивается только при $t = t_9^{\text{П}}$.

В табл. 6.5 приведены результаты настройки элементарного перцептрона при тех же исходных данных, но с помощью γ-системы подкрепления. Во втором столбце табл. 6.5 при $t = t_0$ приведены исходные веса связей и величины сигналов на входе *R*-элемента при предъявлении изображений букв Н и П, а также сумма Σ_1 весов всех связей между *R*- и *A*-нейронами. Значения весов связей в третьем столбце таблицы получены после предъявления изображения буквы Н в момент времени $t = t_1^{\text{Н}}$. Так как $\eta = -0{,}1$, $N_{ак} = 4$ и $N = 6$, то, используя соотношение (6.17) для расчета приращения весов активных связей, получим:

$$\Delta w_1 = \Delta w_2 = \Delta w_3 = \Delta w_4 = \eta - N_{ак}\eta / N = -0{,}1 + 4 \cdot 0{,}1/6 = -0{,}0334, \qquad (6.24)$$

а для приращений весов пассивных связей имеем:

$$\Delta w_5 = \Delta w_6 = -N_{ак}\eta / N = 4 \cdot 0{,}1/6 = 0{,}0667. \qquad (6.25)$$

Таблица 6.5 – Адаптация весов связей перцептрона с помощью γ-системы подкрепления

Весовые коэффициенты и входные сигналы	Моменты времени									
	t_0	$t_1^{\text{Н}}$	$t_2^{\text{П}}$	$t_3^{\text{Н}}$	$t_4^{\text{П}}$	$t_5^{\text{Н}}$	$t_6^{\text{П}}$	$t_7^{\text{П}}$	$t_8^{\text{П}}$	$t_9^{\text{П}}$
w_1^2	0,2	0,1666	0,1333	0,1000	0,0800	0,0600	0,0400	0,0200	0,0000	0,0000
w_2^2	0,8	0,7666	0,7333	0,7000	0,6800	0,6600	0,6400	0,6200	0,6000	0,5750
w_3^2	0,6	0,5666	0,5333	0,5000	0,4800	0,4600	0,4400	0,4200	0,4000	0,3750
w_4^2	0,9	0,8666	0,8333	0,8000	0,7800	0,7600	0,7400	0,7200	0,7000	0,6750
w_5^2	0,8	0,8667	0,9341	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
w_6^2	0,1	0,1667	0,2341	0,3004	0,3804	0,4604	0,5404	0,6204	0,7004	0,7754
$U_{вхR\text{Н}}$	2,5	2,3664	–	2,1000	–	1,9400	–	1,7800	–	1,6250
$U_{вхR\text{П}}$	0,9	–	1,1682	–	1,3800	–	1,5404	–	1,7004	1,7754
Σ_1	3,4	3,3998	3,4004	3,4004	3,4004	3,4004	3,4004	3,4004	3,4004	3,4004

Зная приращения весов связей и используя соотношение:

$$w_i^2(t_1^{\text{Н}}) = w_i^2(t_0) - \Delta w_i\,, \quad i = \overline{1,6},$$

нетрудно получить и численные значения, приведенные в третьем столбце табл. 6.5.

При предъявлении изображения буквы П в момент времени $t = t_2^{\text{П}}$ активными являются только нейроны A_5 и A_6, поэтому $N_{ак} = 2$ и соотношение (6.17) дает следующие численные значения приращений весов связей

$$\Delta w_5 = \Delta w_6 = \eta - N_{ак}\eta / N = 0{,}1 - 2 \cdot 0{,}1/6 = 0{,}0667; \qquad (6.26)$$

$$\Delta w_1 = \ldots = \Delta w_4 = -N_{ак}\eta / N = -2 \cdot 0{,}1/6 = -0{,}0333. \qquad (6.27)$$

Зная приращения Δw_i $(i = \overline{1,6})$ и используя выражение:

$$w_i^2(t_2^{\Pi}) = w_i^2(t_1^{\text{Н}}) + \Delta w_i\,, \quad i = \overline{1,6}\,,$$

несложно получить данные четвертого столбца табл. 6.5.

При расчете приращений весов связей при $t = t_3^{\text{Н}}$ по соотношениям (6.24), (6.25) оказывается, что приращение Δw_5 больше, чем возможно изменение веса связи w_5^2

$$\Delta w_5 = N_{ак}\eta / N = 4 \cdot 0{,}1/0{,}6 = 0{,}0667 > 1 - w_5^2(t_2^{\Pi}) = 1 - 0{,}9341 = 0{,}0659\,.$$

Поэтому для выполнения условия консервативности относительно суммы Σ_1 весов связей необходимо величину разности

$$\Delta_1 w_5 = \Delta w_5 - \left|1 - w_5^2(t_2^{\Pi})\right| = 0{,}0667 - 0{,}0659 = 0{,}0008$$

использовать для изменения весов связей, которые не приняли граничных значений. Один из возможных способов использования разности $\Delta_1 w_5$ – изменить каждую из таких $(N-1)$ активных и пассивных связей на величину:

$$\eta_1 = \frac{\Delta_1 w_5}{N-1}.$$

Значения весов связей при $t = t_3^{\text{Н}}$ приведены в пятом столбце табл. 6.5.

При расчете весов связей с помощью выражений (6.26), (6.27) для $t = t_4^{\Pi}$ необходимо учитывать, что изменяться может вес только одной активной связи:

$$\Delta w_5 = 0;$$

$$\Delta w_6 = \eta - \frac{N_{ак} - 1}{(N-1)}\eta = 0{,}1 - 1 \cdot 0{,}1/5 = 0{,}0800;$$

$$\Delta w_1 = \ldots = \Delta w_4 = (N_{ак} - 1)\eta/(N-1) = -1 \cdot 0{,}1/5 = -0{,}0200.$$

Аналогично при $t = t_5^{\text{Н}}$ имеем

$$\Delta w_1 = \ldots = \Delta w_4 = -\eta + N_{ак}\eta/(N-1) = -0{,}1 + 4 \cdot 0{,}1/5 = -0{,}0200; \quad (6.28)$$

$$\Delta w_5 = 0;$$

$$\Delta w_6 = N_{ак}\eta/(N-1) = 4 \cdot 0{,}1/5 = 0{,}0800.$$

Аналогичным образом рассчитываются веса связей при $t = t_6^{\text{П}}, t_7^{\text{Н}}, t_8^{\text{П}}$. При $t = t_9^{\text{Н}}$ число активных *A*-нейронов при предъявлении изображения буквы Н будет равно четырем, но в выражении (6.28) необходимо использовать $N_{ак} = 3$, поскольку весовой коэффициент $w_1^2 = 0$. Уменьшается до четырех и число весов связей, которые используются для обеспечения постоянства суммы Σ_1 весов всех изменяемых связей перцептрона. В результате получаем:

$$\Delta w_1 = \Delta w_6 = 0;$$

$$\Delta w_2 = \Delta w_3 = \Delta w_4 = -\eta + (N_{ак} - 1)\eta/(N-2) = -0{,}1 + 3 \cdot 0{,}1/4 = -0{,}025;$$

$$\Delta w_5 = (N_{ак} - 1)\eta/(N-2) = 3 \cdot 0{,}1/(6-2) = 0{,}075;$$

$$U_{вх.R\text{Н}} = 1{,}6250 < \theta_R = 1{,}7, \quad U_{вх.R\text{П}} = 1{,}7754 > \theta_R = 1{,}7.$$

Таким образом, при предъявлении буквы Н на входе *R*-элемента сигнал меньше величины порога θ_R и, следовательно, на выходе *R*-нейрона будет требуемый сигнал “–1”, а при предъявлении изображения буквы П на выходном нейроне появится заданный сигнал “+1”.

Как следует из соотношения (6.17) в γ-системе подкрепления при одной и

той же величине сигнала подкрепления η, что и в α-системе, приращение Δw_{ij} веса корректируемой активной связи меньше, чем в α-системе. В связи с этим можно ожидать, что в общем случае процесс настройки нейронной сети вторым методом требует большего числа итераций. Однако анализ данных табл. 6.5 и 6.6 рассматриваемого примера показывает, что число итераций при использовании γ-системы подкрепления не больше, чем при применении α-системы. Это объясняется следующим. Если веса всех связей далеки от граничных значений, то общая сумма S_α изменения весов в α-системе связана только с активными связями:

$$S_\alpha = \eta N_{ak}.$$

В γ-системе аналогичная сумма S_γ может быть равна, меньше или больше S_α, так как на каждой итерации корректируются веса всех связей:

$$S_\gamma = (\eta - \frac{N_{ак}}{N})N_{ак} + \frac{N_{ак}\eta}{N}(N - N_{ак}).$$

С помощью первого слагаемого в этом выражении подсчитывается сумма изменения весов активных связей, а с помощью второго – сумма изменения весов пассивных связей. После преобразований имеем:

$$S_\gamma = N_{ак}\eta + (N_{ак} - \frac{2N_{ак}^2}{N})\eta.$$

Из анализа выражения следует, что в зависимости от соотношения величин $N_{ак}$ и N возможны три выражения:

$$S_\gamma = S_\alpha, \text{ если } N_{ак} = 0{,}5N;$$
$$S_\gamma > S_\alpha, \text{ если } N_{ак} < 0{,}5N;$$

$$S_\gamma < S_\alpha, \text{ если } N_{ак} > 0{,}5N .$$

Из соотношений можно сделать вывод, что в общем случае по числу итераций в процессе обучения элементарного перцептрона ни одна из рассматриваемых систем подкрепления не имеет заметного преимущества.

Розенблатт доказал целый ряд теорем о свойствах элементарных перцептронов. Первая теорема Розенблатта [13] доказывает существование элементарного перцептрона, способного выполнить любую классификацию заданного множества черно-белых изображений, т.е. она показывает, что перцептрон является универсальным устройством для решения любой задачи классификации изображений.

Теорема 6.1. Пусть дано множество $W = \{W_1, ..., W_n\}$ черно-белых изображений на некоторой сетчатке $S = \{S_1, ..., S_m\}$, тогда для любой классификации $C(W)$ множества W черно-белых изображений на два подмножества W^1, W^2 существует не пустой класс элементарных перцептронов с сетчаткой S, способных выполнить эту классификацию.

Доказательство. Для доказательства достаточно показать существование хотя бы одного элементарного перцептрона, способного выполнить произвольную классификацию $C(W)$. Рассмотрим перцептрон, каждому изображению W_k $(k = \overline{1, n})$ на сетчатке S которого соответствует один A-элемент – нейрон A_k, функционирование которого определяется выражением:

$$U_{вых.Ak} = \begin{cases} 1, \text{ если } U_{вх.Ak} \geq \theta, \\ 0, \text{ если } U_{вх.Ak} < \theta, \end{cases} \tag{6.29}$$

где $U_{вых.Ak}$ – выходной сигнал нейрона A_k; $U_{вх.Ak}$ – сигнал на входе нейрона A_k.

$$U_{вх.Ak} = \sum_{j=1}^{m} U_{вых.Sj} w_{jk}; \tag{6.30}$$

$U_{вых.Sj}$ – выходной сигнал j-го S-элемента,

$$U_{вых\,Sj} = \begin{cases} 1, \text{ если } S\text{-элемент возбужден}, \\ -1, \text{если } S\text{-элемент заторможен}, \end{cases} \tag{6.31}$$

w_{jk} – вес связи между j-м S-элементом и k-м A-нейроном; θ – порог срабатывания k-го A-элемента; $\theta = m$.

Для каждого изображения W_k зададим веса w_{jk} ($j = \overline{1,m},\ k = \overline{1,n}$) соотношением:

$$w_{jk} = \begin{cases} 1, \text{если элемент } S_j \text{ возбужден } k\text{-м изображением}, \\ -1, \text{если элемент } S_j \text{ заторможен } k\text{-м изображением}. \end{cases} \tag{6.32}$$

При предъявлении любого изображения W_k ($k = \overline{1,n}$) перцептрону, удовлетворяющему соотношениям (6.29) – (6.32), только на входе одного k-го A-нейрона, будет сигнал, равный в соответствии с соотношением (6.30) числу m, и только на выходе этого нейрона в соответствии с выражением (6.29) будет единичный выходной сигнал. Теперь для правильного выполнения разделение исходного множества W на два подмножества W^1, W^2 с помощью элементарного перцептрона необходимо только всем весам связей между R- и A-нейронами, которые соответствуют A-элементам, возбуждаемым изображениями подмножества W^1, придать положительные значения, а всем весам связей A-нейронов, которые возбуждаются изображениями подмножества W^2, – отрицательные значения, а затем задать выходной сигнал R-нейрона $U_{вых.R}$ выражением вида:

$$U_{вых.R} = \begin{cases} 1, \text{если } U_{вх.R} \geq 0, \\ -1, \text{если } U_{вх.R} < 0, \end{cases}$$

где $U_{вх.R}$ – входной сигнал R-элемента.

В этом случае все изображения подмножества W^1 будут кодироваться по-

ложительным единичным выходным сигналом нейронной сети, а подмножества W^2 – отрицательным, т.е. будет правильно выполняться классификация $C(W)$ исходного множества W изображений.

Хотя построенная таким образом нейронная сеть не имеет существенного практического значения, однако ее наличие показывает, что элементарный перцептрон является универсальным устройством классификации любого множества изображений на два класса. В том случае, когда число изображений множества W превышает число A-нейронов, элементарный перцептрон теряет свою универсальную способность классифицировать изображения .

Теорема 6.2. Если число n изображений в множестве W больше числа A-элементов элементарного перцептрона, то существует некоторая классификация $C(W)$ множества W черно-белых изображений на два подмножества W^1, W^2, которая не может быть выполнена перцептроном.

Теорема 6.1 доказывает существование элементарного перцептрона, способного выполнять любую заданную классификацию $C(W)$ изображений некоторого множества W на два класса, однако она не указывает алгоритмов достижения этой способности в процессе обучения нейронной сети. Розенблаттом была доказана важная для теории элементарных перцептронов теорема о наличии таких алгоритмов для α-перцептронов.

Рассмотрим некоторую классификацию $C(W)$ множества W изображений на два подмножества W^1, W^2, которая может осуществляться перцептроном с R-элементом, выходной сигнал которого удовлетворяет условиям:

$$U_{вых.R} = \begin{cases} 1, \text{если } W_k \in W^1, \\ -1, \text{если } W_k \in W^2, \end{cases} \quad k = \overline{1,n}. \tag{6.33}$$

Положим также, что:

$$\rho_k = \begin{cases} 1, \text{если } W_k \in W^1, \\ -1, \text{если } W_k \in W^2, \end{cases} \quad k = \overline{1,n}. \tag{6.34}$$

Определение 6.9. Метод коррекции ошибок без квантования – это метод системы подкрепления с коррекцией ошибок, когда при ошибочной реакции *R*-элемента с порогом θ на некоторое изображение $W_k \in W$ к весу каждой из связей, соединяющих активные *A*-нейроны с *R*-элементом, прибавляется величина $\eta = \rho_k R_k$, где коэффициент R_k выбирается из условия, что после коррекции весов связей выполняется соотношение (6.33), т.е. перцептрон правильно классифицирует предъявленное изображение. В методе коррекции ошибок с квантованием применяется это же правило коррекции весов связей, но величина R_k в общем случае гораздо меньше и правильный сигнал на выходе *R*-нейрона, как правило, достигается не за одну итерацию.

Теорема 6.3. Пусть дан элементарный α-перцептрон, множество черно-белых изображений $W = \{W_1, ..., W_n\}$, некоторая классификация *C*(*W*) этих изображений на два подмножества, которая может быть выполнена α-перцептроном. Изображения $W_1, ..., W_n \in W$ подаются на вход перцептрона в произвольной последовательности, в которой каждое из них появляется неоднократно, через конечное число предъявлений других изображений. Тогда процесс обучения перцептрона методом коррекции ошибок (с квантованием или без квантования подкрепления) независимо от начальных значений весов связей между *R*- и *A*-элементами всегда приводит за конечное число итераций к множеству весов связей, с помощью которых α-перцептрон может выполнить заданную классификацию изображений.

Теорема 6.3 доказывает наличие сходящегося детерминированного метода обучения с коррекцией ошибок для элементарного перцептрона с квантованием или без квантования подкрепления. Следующая теорема доказывает, что обучение элементарного перцептрона может быть выполнено и при менее жестких требованиях к виду коррекции – методом коррекции ошибок со случайным законом подкрепления, когда при появлении ошибки сигнал подкрепления формируется как в α-системе, но знак подкрепления с вероятностью 0,5 может быть положительным или отрицательным.

Теорема 6.4. Пусть дан элементарный α-перцептрон с конечным числом значений весов связей между *R*- и *A*-нейронами, множество черно-белых изображений $W = \{W_1, ..., W_n\}$, некоторая классификация $C(W)$ этих изображений на два подмножества, которая может быть выполнена α-перцептроном при некотором наборе весов связей между *R*- и *A*-нейронами, изображения $W_1, ..., W_n \in W$ подаются на вход перцептрона в произвольной последовательности, в которой каждое из них появляется неоднократно через конечное число предъявлений других изображений. Тогда процесс обучения перцептрона, величина сигналов подкрепления которого формируется как и в α-системе с квантованием подкрепления, а знак подкрепления выбирается с вероятностью 0,5 положительным или отрицательным, может быть выполнен за конечное время с вероятностью, равной единице, независимо от начальных значений весов связей между *R*- и *A*-нейронами.

Естественно, что метод со случайным знаком подкрепления требует большего объема вычислений при обучении перцептрона, чем прямая коррекция ошибок с квантованием или без квантования подкрепления. Еще большего объема вычислений требует метод, в котором производится случайный выбор не только знака, но и величины подкрепления. Розенблаттом доказана теорема о том, что с вероятностью, равной единице, обучение перцептрона может быть выполнено за конечное время и методом коррекции случайными возмущениями, когда подкрепление формируется как в α-системе, но при этом величина η и знак подкрепления для каждого веса связи выбираются отдельно и независимо в соответствии с некоторым заданным законом распределения вероятностей.

Менее универсальной системой подкрепления, чем α-система и системы со случайным формированием подкрепления, является γ-система. Это доказывает следующая теорема Розенблатта.

Теорема 6.5. Пусть дан элементарный γ-перцептрон, подмножество $W = \{W_1, ..., W_n\}$ черно-белых изображений и некоторая классификация $C(W)$

этих изображений на два класса W^1, W^2. Тогда для выполнения классификации $C(W)$ может существовать набор весов связей, недостижимый для γ-системы подкреплений.

6.7. Трехслойные перцептроны с несколькими *R*-элементами и переменными весами связей между *S*- и *A*-нейронами

Естественным обобщением элементарного перцептрона являются перцептроны, отличающиеся от элементарного только наличием нескольких *R*-нейронов. Такие нейронные сети могут иметь три разновидности топологической структуры. Для первой структуры характерно соединение каждого *A*-элемента с каждым *R*-нейроном. Для второй присуще разделение множества *A*-элементов на непересекающиеся подмножества, каждое из которых связано только с одним *R*-элементом. Эта структура представляет собой как бы объединение нескольких элементарных перцептронов, имеющих одно общее множество *S*-элементов. Для третьей топологической структуры характерно соединение каждого *R*-элемента с некоторым подмножеством случайно выбранных *A*-элементов.

Перцептрон с *NR*-элементами, каждый из которых имеет два состояния: +1 и –1 или +1 и 0, может быть использован для распознавания от *N* до 2^N образов. Число распознаваемых образов зависит от способа их кодирования *R*-элементами. Если используется позиционное кодирование, когда каждому образу назначается положительный выход только одного *R*-элемента (и нулевой или отрицательный выход всех остальных *R*-нейронов), то перцептрон может быть обучен распознаванию *N* образов. Если же каждому образу в соответствие ставится произвольный двоичный код на выходных нейронах перцептрона, т.е. используется конфигурационное кодирование образов, то перцептрон может быть обучен распознаванию до 2^N образов.

При наличии возможности выбора между двумя видами кодирования,

лучше выбрать позиционный код, так как он при прочих равных условиях во многих случаях приводит к лучшим результатам, хотя по числу используемых *R*-элементов и является менее экономичным. Если число N_1 образов превышает число *NR*-нейронов, то необходимо использование конфигурационного кода. При этом желательно, чтобы N_1 не было близко к 2^N, так как при предельных значениях числа распознаваемых образов выбор кода отдельных образов может стать решающим фактором для успешного обучения перцептрона.

Для обучения перцептронов с несколькими *R*-элементами может использоваться модифицированный метод коррекции ошибок. Как и для элементарного перцептрона, в этом случае, прежде всего, определяется наличие неправильной реакции нейронной сети на предъявленное изображение. Это может выполняться с помощью соотношения вида:

$$\eta = R - r, \tag{6.34}$$

где $\eta = (\eta_1, ..., \eta_N)$ – вектор подкреплений весов связей от *A*- к *R*-нейронам; η_i $(i = \overline{1, N})$ – подкрепление весов связей, идущих к *i*-му *R*-элементу; $R = (R_1, ..., R_N)$ – вектор желаемых реакций *R*-нейронов; $r = (r_1, ..., r_N)$ – вектор действительных реакций выходных нейронов.

Если $\eta = 0$, т.е. реакция перцептрона правильна, то веса связей между *A*- и *R*-нейронами остаются неизменными. Если $\eta \neq 0$, то с помощью скалярных компонент векторного соотношения (6.34) выполняется коррекция только весов связей, идущих к *R*-элементам, реакция которых отличается от желаемой.

Система подкрепления на основе соотношения (6.34) всегда приводит к настройке перцептрона, если такая настройка вообще существует для данного множества изображений, способа их кодирования и матрицы постоянных весов *S*-*A*-связей.

Условие фиксированности значений весов связей между двумя первыми слоями нейронов может негативно сказываться на возможностях перцептрона. В связи с этим в процессе его обучения желательна оптимизация весов связей

не только между *A*- и *R*-элементами, но и между *S*- и *A*-нейронами.

В начале 60-х годов Розенблатт предложил несколько методов коррекции *S*-*A*-связей, однако они или не гарантировали определения оптимального решения, или требовали чрезмерных вычислительных затрат даже для нейронных сетей с небольшим числом нейронов, поскольку использовали случайный поиск для решения задачи, относящейся к классу *NP*-сложных. (Задача относится к классу *NP*-сложных, если не существует алгоритма, который бы мог решить ее за время, растущее с размерностью задачи не быстрее полинома конечной степени). Существенный прорыв в обучении трех- и многослойных перцептронов был получен только после появления метода обратного распространения ошибки [15, 16], который будет рассмотрен в дальнейшем.

Перцептроны с двоичными *S*-, *A*- и *R*-элементами нейронами способны распознавать только линейно разделимые классы изображений.

Последний факт был установлен в 1969 году в работе Минского и Пейперта “Перцептроны”, в которой авторы сделали глубокий теоретический анализ трудностей применения трехрядных перцептронов. Они показали, что затруднения в использовании перцептронов для распознавания образов носят принципиальный характер, связанный с тем, что этот тип нейронных сетей может распознавать только линейно разделимые классы изображений. Для изображений на плоскости это означает, что в качестве разделителя является прямая линия (рис.6.9), для изображений в трехмерном пространстве – плоскость, для четырех и более многомерных пространств – гиперплоскость.

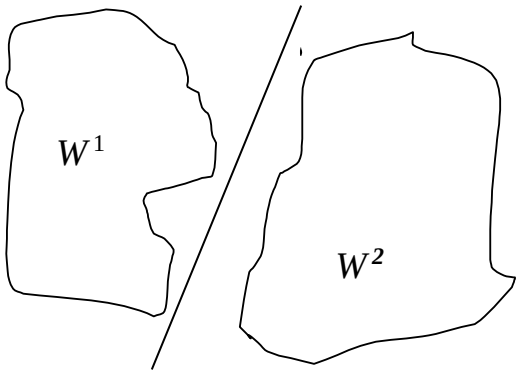


Рис. 6.9. Линейно разделимые классы изображений

Перцептрон с n двоичными чувствительными S-нейронами может воспринимать 2^n различных входных изображений. Поскольку при одном R-элементе каждому входному изображению может соответствовать два различных выхода, то с помощью элементарного перцептрона может быть получено $n_1 = 2^{2^n}$ различных функций от n двоичных переменных или n_1 различных классификаций 2^n входных изображений.

В табл. 6.6 для различных значений n приведено общее число n_1 функций и число n_2 классификаций, соответствующих линейно разделимым изображениям, а также процентное содержание $\delta = 100n_2/n_1$ линейно выполнимых классификаций в общем числе возможных.

Таблица 6.6 – Линейно разделимые и неразделимые функции

n	n_1	n_2	δ
1	4	4	100
2	16	14	87,5
3	256	104	40,6
4	65536	1882	2,87
5	$4{,}295 \cdot 10^{9}$	94572	0,0022
6	$1{,}845 \cdot 10^{19}$	15028134	$0{,}0081 \cdot 10^{-10}$

Из анализа данных четвертого столбца табл. 6.6 следует, что при $n \geq 5$ доля линейно разделимых изображений среди их общего числа ничтожно мала. В связи с этим область возможного применения перцептронов с двоичными S-, A- и R-элементами как устройств распознавания очень ограничена. И она еще более сужается из-за отсутствия простого и универсального способа определения линейной разделимости конкретных образов.

Рассмотренные и другие подобные примеры Минского и Пейперта привели к резкому падению интереса исследователей к двух- и трехрядным перцептронам, а поскольку в то время отсутствовали эффективные методы обуче-

ния многорядных перцептронов с нелинейными функциями активации, то и ко всем перцептронам вообще.

Практическое использование перцептронов для распознавания образов стало возможно только в 80-е годы XX века, когда для настройки или обучения нейронных сетей появились эффективные алгоритмы метода обратного распространения ошибки (back-propagation error).

6.8. Метод обратного распространения ошибки

Метод обратного распространения ошибки, для названия которого очень часто используют только первые три слова, сыграл важную роль в возрождении интереса к нейронным сетям как инструменту решения широкого круга разнообразных проблем. Метод был разработан независимо друг от друга несколькими исследователями в период с 1974 по 1986 годы [28], но получил широкое признание и применение только после публикации Румельхарта, Хинтона и Вильямса в 1986 году [28]. Метод обратного распространения фактически представляет собой обобщение метода наименьших квадратов применительно к многослойным нейронным сетям. Он с помощью простого градиентного спуска минимизирует среднеквадратичную ошибку между действительным и требуемым выходом нейронной сети.

6.8.1. Требования к функциям активации в методе обратного распространения ошибки.

Использование градиентного спуска предполагает определенные требования к активационным функциям нейронов. Функция активации должна быть непрерывной, дифференцируемой и монотонно возрастающей, аппроксимировать единичный максимум и минимум. Ее производная должна просто вычисляться и желательно, чтобы она выражалась через значения функции. (Требова-

ние монотонности возрастания для радиально-симметричной функции (6.10) выполняется только на интервале (–∞, 0), а на интервале (0, ∞), оно заменяется требованием монотонности убывания.)

Одной из наиболее распространенных является бинарная сигмоидальная функция активации (6.8) с областью значений (0, 1) и областью определения (–∞, ∞)

$$g(x) = \frac{1}{1+e^{-x}}.$$

При стремлении x к –∞ функция $g(x)$ стремится к нулю, а при $x \to \infty$ она стремится к единице.

Производная функции $g(x)$ равна

$$g'(x) = \frac{e^{-x}}{(1+\mathrm{e}^{-x})^2} = g(x)(1-g(x)). \qquad (6.35)$$

Поскольку ни один из сомножителей выражения (6.35) не обращается в нуль в интервале (–∞, ∞), то функция $g(x)$ не имеет экстремальных точек, а так как в рассматриваемом интервале оба сомножителя неотрицательны, то $g'(x) > 0$. Следовательно, функция $g(x)$ в интервале (–∞, ∞) монотонно возрастает от нуля до единицы, поскольку

$$\lim_{x\to-\infty}(g(x)) = 0 \quad \text{и} \quad \lim_{x\to\infty}(g(x)) = 1.$$

Функция $g'(x)$ при стремлении x к –∞ или к ∞ имеет своим пределом нуль. Ее производная

$$(g'(x))' = \frac{e^{-x}(-1+e^{-x})}{(1+e^{-x})^3}$$

положительна при $x \in (-\infty, 0)$, отрицательна при $x \in (0, \infty)$ и обращается в нуль только при $x = 0$. Функция $g'(x)$ при $x = 0$ имеет единственный максимум:

$$g'(0) = \frac{e^0}{(1+e^0)^2} = \frac{1}{4}.$$

Производная функции $g(x)$ симметрична относительно оси ординат, т.е.

$$g'(x) = g'(-x)$$

или

$$\frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^x}{(1+e^x)^2}.$$

Действительно

$$\frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{\left(\frac{e^x+1}{e^x}\right)^2} = \frac{e^x}{(1+e^x)^2}.$$

Для функции $g'(x)$ справедливо также свойство

$$\int_{-\infty}^{\infty} g'(x)dx = \frac{1}{1+e^{-x}}\Bigg|_{-\infty}^{\infty} = 1. \qquad (6.36)$$

Если рассматривать сигмоидальную функцию активации

$$g(x) = \frac{1}{1 + e^{-kx}}$$

при k, стремящемся к ∞, то получим функцию активации бинарного нейрона (соотношение (6.2) и рис. 6.2, а). При этом производная $g'(x)$ сигмоидальной функции в силу соотношения (6.2) и наличия единственного максимума при $x = 0$ превращается в δ-функцию Дирака (или просто в δ-функцию), т.е. при $k \to \infty$, из сигмоидальной функции можно получить пороговую функцию активации простейшего бинарного нейрона. Сигмоидальная характеристика обладает еще одним важным достоинством, следующим из свойств ее первой производной (или тангенса угла наклона касательной к графику функции $g(x)$). Для слабых сигналов эта характеристика вход-выход обладает наибольшим усилением, так как $g'(x)$ принимает максимальное значение при $x = 0$. Когда же величина положительного или отрицательного сигнала возрастает, то усиление падает, причем тем больше, чем больше величина входного сигнала отличается от нуля. Это связано с тем, что функция $g'(x)$ монотонна и симметрична относительно оси ординат и убывает до нуля при $x \to \pm\infty$.

Таким образом, благодаря свойствам сигмоидальной кривой автоматически регулируется усиление нейронов сети и решается проблема обработки как слабых, так и сильных сигналов.

Другой распространенной функцией активации является биполярная сигмоидальная функция (6.9), которая имеет область значений (–1, 1) и определяется как

$$g_1(x) = 2g(x) - 1 = \frac{2}{1 + e^{-kx}} - 1.$$

Ее производная равна

$$g_1'(x) = \frac{k}{2}(1 - g_1(x))(1 + g_1(x)).$$

Поскольку

$$g_1(x) = \frac{2}{1+e^{-kx}} - 1 = \frac{1-e^{-kx}}{1+e^{-kx}},$$

а гиперболический тангенс определяется выражением

$$\text{th}x = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

которое можно преобразовать к виду

$$\text{th}x = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}},$$

то эту активационную функцию можно рассматривать и как гиперболический тангенс.

Метод обратного распространения ошибки может использоваться и для настройки нейронных сетей, элементы которых имеют не сигмоидальные функции активации. Примером такой функции является радиально-симметричная

$$g_2(x) = e^{-kx^2}.$$

Ее производная определяется соотношением

$$g_2'(x) = -2kxe^{-kx^2} = -2kxg_2(x).$$

6.8.2. Основной алгоритм метода обратного распространения ошибки.

Метод обратного распространения ошибки разрабатывался для обучения нейронных сетей с любым числом слоев. Однако для изложения сути метода достаточно рассмотреть сеть, изображенную на рис. 6.10. В настоящее время в

литературе нет единого подхода к тому, как считать число слоев многослойных нейронных сетей. Одни авторы подсчитывают число слоев нейронов, включая и входной слой, другие – только число слоев нейронов с множеством присоединенных к их входам связей с весовыми коэффициентами, утверждая, что входные нейроны не образуют полноценного слоя, поскольку выполняют только функцию распределения входных сигналов. При первом подходе сеть, изображенная на рис. 6.10, имеет три слоя, а при втором – два. Мы придерживались и будем придерживаться первого подхода, считая, что нейронная сеть, изображенная на рис. 6.10, имеет три слоя.

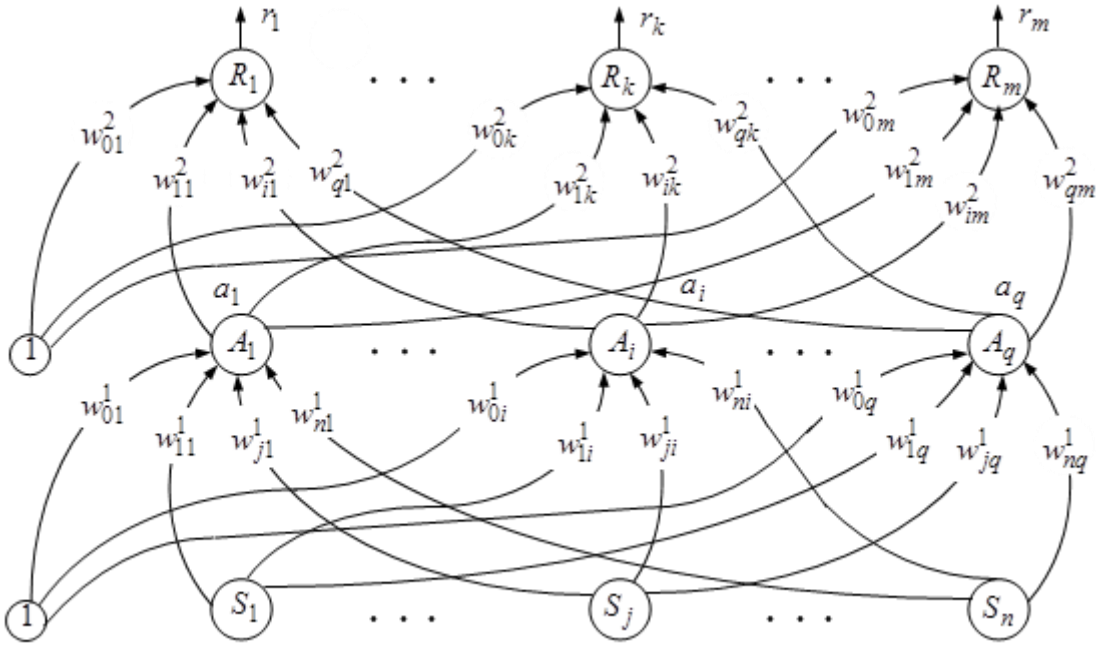


Рис. 6.10. Трехслойная сеть для иллюстрации метода обратного распространения ошибки

На рис. 6.10 показано, что выходные *R*-нейроны и нейроны скрытого слоя (*A*-элементы) могут иметь смещения, которые действуют подобно нейронам, у которых выходной сигнал всегда равен единице (обычно эти элементы нейронных сетей, изображенные на рис. 6.10, в явном виде не показывают).

Смещения на типичные элементы R_k, A_i выходного и скрытого слоя подаются с помощью связей, имеющих соответственно весовые коэффициенты

w_{0k}^2, w_{0i}^1 $(k=\overline{1,m},\ i=\overline{1,q})$. В силу единичных сигналов на входах связей величины смещений на указанные нейроны также равны соответственно w_{0k}^2, w_{0i}^1 $(k=\overline{1,m},\ i=\overline{1,q})$.

Входные изображения $S^1=(S_1^1,...,S_j^1,...,S_n^1)$, ..., $S^p=(S_1^p,...,S_j^p,...,S_n^p)$, ..., $S^L=(S_1^L,...,S_j^L,...,S_n^L)$ подаются на чувствительные нейроны первого слоя $(S_j,\ j=\overline{1,n})$, которые не преобразовывают сигналы входных изображений

$$S_j^p=U_{вх.Sj}^p,\ \ j=\overline{1,n},\ \ p=\overline{1,L},$$

а только распределяют их по входам нейронов скрытого слоя, т.е.

$$U_{вых.Sj}^p=U_{вх.Sj}^p,\ \ j=\overline{1,n},\ \ p=\overline{1,L},$$

где $U_{вых.Sj}^p$, $U_{вх.Sj}^p$ – соответственно входные и выходные сигналы нейронов $S_j,\ j=\overline{1,n}$.

Каждому входному изображению S^p $(p=\overline{1,L})$ ставится в соответствие вектор $t^p=(t_1^p,...,t_k^p,...,t_m^p)$, задающий требуемый выход нейронной сети. Входные изображения S^p и соответствующие им вектора t^p образуют обучающие пары $(S^p,\ t^p)$, $p=\overline{1,L}$ нейронной сети.

Перед строгим описанием конкретного алгоритма поясним общую идею метода обратного распространения ошибки.

Метод обратного распространения имеет три явно выраженных этапа или фазы: прямую передачу входного обучающего изображения, обратное распространение ошибки (если она имеется) и коррекцию весов связей (если имела место ошибка).

В течение прямой передачи каждый входной нейрон S_j ($j=\overline{1,n}$) принимает сигнал от предъявленного изображения $S^p=(S_1^p,...,S_j^p,...,S_n^p)$ и передает этот сигнал каждому элементу A_i ($i=\overline{1,q}$) скрытого слоя. Каждый скрытый нейрон суммирует все свои входные сигналы и с помощью функции активации вычисляет выходной сигнал (a_i), который посылает каждому *R*-элементу. Каждый элемент R_k ($k=\overline{1,m}$) вычисляет свой выходной сигнал r_k. Вектор (r_1 , ..., r_k , ..., r_m) выходных сигналов *R*-элементов является реакцией сети на данное входное изображение. На втором этапе работы алгоритма каждый выходной элемент R_k ($k=\overline{1,m}$) сравнивает свой выходной сигнал с компонентами требуемого выходного вектора $t^p=(t_1^p,\ ...,\ t_k^p,\ ...,\ t_m^p)$ нейронной сети для данного входного изображения S^p с целью определения величины своей ошибки. Основываясь на рассчитанной ошибке, для каждого *R*-элемента вычисляется величина δ_k ($k=\overline{1,m}$), которая используется для распространения ошибки от выходного нейрона R_k назад ко всем нейронам скрытого слоя, соединенных с R_k. Позднее δ_k ($k=\overline{1,m}$), используются также для коррекции весов связей между скрытым и выходным слоем. Подобным образом для каждого скрытого элемента A_i ($i=\overline{1,q}$) вычисляется величина δ_i ($\mathrm{i}=\overline{1,q}$). Множество величин δ_i ($i=\overline{1,q}$) используется для коррекции весов связей между входным и скрытым слоем.

На третьем этапе работы алгоритма с помощью величин δ_k ($k=\overline{1,m}$), δ_i ($\mathrm{i}=\overline{1,q}$) производится коррекция весов связей w_{ik}^2 и w_{ji}^1, а также смещений w_{0k}^2, w_{0i}^1 нейронов выходного и скрытого слоев.

Условий окончания работы алгоритма должно быть не менее двух – для успешного и неудачного его применения. Примером условия для успешного окончания работы алгоритма может служить следующее:

Алгоритм оканчивает работу, если при предъявлении каждого изображения S^p ($p=\overline{1,L}$) без коррекции весов связей и смещений на выходе нейронной сети появляется вектор $r^p=(r_1^p,...,r_k^p,...,r_m^p)$, удовлетворяющий требованиям

$$\left| t_k^p - r_k^p \right| \le E_k^p,\ k=\overline{1,m},$$

где t_k^p, E_k^p – соответственно компоненты целевого вектора $t^p=(t_1^p,...,t_k^p,...,t_m^p)$, и вектора $E^p=(E_1^p,...,E_k^p,...,E_m^p)$ допустимой ошибки для *p*-го изображения.

Условием прекращения работы алгоритма при его неудачном применении может служить достижение алгоритмом предельного времени T_n обучения нейронной сети.

Обучающий алгоритм метода обратного распространения ошибки

Шаг 1. Множеством небольших случайных значений инициируются веса всех связей нейронной сети.

Шаг 2. Проверяются условия корректности задания исходных данных и необходимости работы алгоритма и, если они выполняются, то реализуются шаги 3 – 11 алгоритма.

Шаг 3. Для каждой обучающей пары (входное изображение $S^P=(S_1^p,...,S_n^p)$, необходимый выходной вектор $t^p=(t_1^p,\ ...,t_m^p)$) выполняются шаги 4 – 10 алгоритма.

Этап прямой передачи

Шаг 4. Каждый входной чувствительный нейрон S_j ($j=\overline{1,n}$) получает входной сигнал S_j^p и передает его ко всем элементам A_i ($i=\overline{1,q}$) скрытого слоя.

Шаг 5. Каждый скрытый нейрон A_i $(i=\overline{1,q})$ суммирует свои взвешенные входные сигналы:

$$U_{вх.Ai} = w_{0i}^1 + \sum_{j=1}^{n} S_j^p w_{ji}^1,$$

и с помощью своей функции активации g_{Ai} рассчитывает выходной сигнал

$$a_i = U_{вых.Ai} = g_{Ai}(U_{вх.Ai}),$$

и посылает его на входы всех *R*-элементов.

Шаг 6. Каждый выходной нейрон R_k $(k=\overline{1,m})$ суммирует свои взвешенные входные сигналы:

$$U_{вх.Rk} = w_{0k}^2 + \sum_{i=1}^{q} a_i w_{ik}^2,$$

и с помощью своей функции активации рассчитывает выходной сигнал

$$r_k = U_{вых.Rk} = g_{Rk}(U_{вх.Rk}).$$

Этап обратного распространения ошибки

Шаг 7. Каждый выходной нейрон R_k $(k=\overline{1,m})$ получает компоненту t_k^p требуемого выходного вектора $t^p = (t_1^p, \ldots, t_m^p)$, вычисляет свою ошибку

$$E_k = \frac{1}{2}(t_k^p - r_k)^2 \frac{1}{2} e_k^2 = \frac{1}{2}(t_k^p - g_{Rk}(U_{вх.Rk}))^2 =$$

$$= \frac{1}{2}(t_k^p - g_{Rk}(\sum_{i=1}^{q} a_i w_{ik}^2 + w_{0k}^2))^2$$

и ее частные производные

$$\delta_k = \frac{\partial E_k}{\partial w_{ik}} = \frac{\partial E_k}{\partial e_k}\frac{\partial e_k}{\partial U_{вых Rk}}\frac{\partial U_{вых Rk}}{\partial U_{вхRk}}\frac{\partial U_{вхRk}}{\partial w_{ik}} = (t_k^p - r_k)(g_{Rk}(U_{вхRk}))' w_{ik}$$

определяет с помощью коэффициента скорости обучения α (обычно α удовлетворяет неравенствам $0{,}01 \le \alpha \le 1$) приращения весов связей w_{ik}^2 ($i = \overline{1,q}$, $k = \overline{1,m}$) и смещений w_{0k}^2 ($k = \overline{1,m}$):

$$\Delta w_{ik}^2 = \alpha \delta_k a_i, \quad \Delta w_{0k}^2 = \alpha \delta_k,$$

и посылает производные δ_k к элементам скрытого слоя.

Шаг 8. Каждый скрытый элемент A_i ($i = \overline{1,q}$) суммирует свои дельта-входы от элементов верхнего слоя и определяет суммарную ошибку на своем входе

$$\delta_{вх.Ai} = U_{вх.Ai} = \sum_{k=1}^{m} \delta_k w_{ik}^2,$$

затем вычисляет дельта-функцию на своем выходе

$$\delta_i = \delta_{вх.Ai} g_i'(U_{вх.Ai})$$

и приращения весов связей w_{ji}^1 ($j = \overline{1,n}$, $i = \overline{1,q}$) и смещений w_{0j}^1 ($j = \overline{1,n}$):

$$\Delta w_{ji}^1 = \alpha \delta_i S_j^p, \quad \Delta w_{0i}^1 = \alpha \delta_i.$$

Этап коррекции весов связей и смещений

Шаг 9. Каждый выходной элемент R_k ($k = \overline{1,m}$) адаптирует свое смещение и веса:

$$w_{ik}^2 = w_{ik}^2 + \Delta w_{ik}^2, \quad i = \overline{0, q}, \; k = \overline{1, m}.$$

Шаг 10. Каждый скрытый элемент A_i $(i = \overline{1, q})$ адаптирует свое смещение и веса:

$$w_{ji}^1 = w_{ji}^1 + \Delta w_{ji}^1, \quad j = \overline{0, n}, \; i = \overline{1, q}.$$

Шаг 11. Проверяются условия останова и, если они не выполняются, то переход к шагу 3 алгоритма.

Шаг 12. Останов.

6.9. Проблемы модификации и обобщения основного алгоритма метода обратного распространения

Один из основных недостатков алгоритмов обратного распространения ошибки – медленная сходимость. Во многих случаях для обучения многослойной сети требуются сотни, тысячи и даже десятки тысяч предъявлений всего обучающего множества изображений.

Другой недостаток основного алгоритма связан с использованием градиентного спуска по поверхности ошибки выхода нейронной сети. Поверхность ошибки даже для относительно простых нейронных сетей, содержащих не более 15 – 20 нейронов, имеет сложный рельеф и состоит из холмов и пиков, оврагов и впадин, складок и долин в пространстве высокой размерности. Алгоритм обратного распространения легко находит точки локальных экстремумов, из которых он не способен выбраться.

Неудачи в обучении сети могут быть связаны и с попаданием в результате коррекции весов связей многих или всех нейронов в области, где производные функции активации очень малы. Поскольку в процессе обучения величины приращений весов связей, используемых для адаптации нейронной сети, про-

порциональны этим производным, то процесс обучения сети может практически остановиться и наступит “паралич сети”.

В связи с отмеченными проблемами многими исследователями были разработаны различные улучшения, модификации и обобщения приведенного выше основного алгоритма метода обратного распространения ошибки [19, 20, 23, 28-32]. Число предложенных алгоритмов слишком велико, чтобы их можно было охватить в учебном пособии. Поэтому остановимся только на некоторых перспективных разработках.

В работах [20, 28] описан алгоритм, использующий для адаптации весов связей нейронных сетей приращения весов, полученных не только на текущем, но и на предшествующем проходе алгоритма:

$$w_{pq}^{i}(t+1) = w_{pq}^{i}(t) + \Delta w_{pq}^{i}(t) + \alpha_1 \Delta w_{pq}^{i}(t-1),$$

где α_1 – постоянный коэффициент, часто выбираемый около 0,9 [20].

Алгоритм с такой коррекцией весов связей позволяет спускаться по дну узких оврагов поверхности ошибки. Он позволяет эффективно получать хорошие решения одних классов задач, но может давать незначительный или даже отрицательный эффект при решении других.

В работах [19, 32] описан подобный алгоритм, основанный на экспоненциальном сглаживании за счет использования в качестве коэффициента α_1 экспоненциально затухающей функции.

Для определения более точных оценок величин приращений весов связей и, следовательно, ускорения сходимости алгоритмов предложен метод обратного распространения ошибки второго порядка, в котором предусмотрены процедуры использования вторых производных. На каждом проходе алгоритма метод требует дополнительного объема вычислений по сравнению с основным алгоритмом, но из-за более точного выбора приращений весов общий объем вычислений, необходимых для обучения нейронных сетей, может быть существенно меньше. Однако более эффективен этот метод при решении далеко не всех задач. Например, при обучении решению задачи ИСКЛЮЧАЮЩЕЕ ИЛИ на од-

ной из сетей методом обратного распространения потребовалось 245 предъявлений обучающего множества изображений и 4986 предъявлений при использовании алгоритма обратного распространения ошибки второго порядка [20].

Для борьбы с попаданием алгоритмов обратного распространения в точки локальных минимумов предложены методы, объединяющие статистические подходы с градиентным спуском в пространстве ошибки. Рассмотрим один из них – метод комбинированного обратного распространения с обучением Коши. Приращения весов, используемых для адаптации нейронной сети, в этом методе состоят из двух компонент:

$$w_{pq}^{i}(t+1) = w_{pq}^{i}(t) + \beta\Delta w_{pq}^{i} + (1-\beta)\Delta w_{pqk}^{i},$$

где $w_{pq}^{i}(t+1)$, $w_{pq}^{i}(t)$ – значения весов связей соответственно в будущий и текущий моменты времени между нейронами i-го и $(i + 1)$-го слоев; Δw_{pq}^{i} – приращение веса, вычисленное с использованием алгоритма обратного распространения; Δw_{pqk}^{i} – приращение веса, определяемое алгоритмом обучения Коши; β – коэффициент, задающий относительный вклад каждого из приращений в общее приращение веса связи; при $\beta = 1$ имеем основной алгоритм обратного распространения, а при $\beta = 0$ – алгоритм обучения Коши.

Статистический алгоритм обучения Коши использует идеи, заимствованные из физических процессов, протекающих при отжиге металлов. Атомы металла, нагретого до температуры, превышающей его точку плавления, находятся в сильном хаотическом движении, которое препятствует возникновению низкоэнергетических состояний. Однако при понижении температуры вероятность появления таких состояний увеличивается, а поскольку каждый атом стремится к состоянию с минимумом энергии, то в конце концов в металле достигается одно из найнизших возможных энергетических состояний. Во время отжига металла распределение различных энергетических уровней описывается выражением

$$P(E) = e^{-E/kT}, \tag{6.37}$$

где $P(E)$ – вероятность нахождения атома металла в состоянии с энергией E; k – постоянная Больцмана; T – температура металла в градусах Кельвина.

Анализ выражения (6.37) показывает, что при высоких значениях температуры вероятность $P(E)$ появления всех энергетических состояний близка к нулю, но по мере снижения температуры вероятность появления низкоэнергетических кристаллических решеток существенно выше по сравнению с высокоэнергетическими состояниями.

По аналогии с описанным физическим процессом отжига металла был предложен алгоритм обучения Коши для нейронных сетей, который предусматривает выполнение следующих шагов:

Шаг 1. Задается высокая искусственная начальная температура T_0, которая с течением времени t уменьшается в соответствии с выражением

$$T(t) = \frac{T_0}{1+t}. \tag{6.38}$$

Шаг 2. Множеством небольших случайных значений инициируются веса всех связей.

Шаг 3. Для каждой обучающей пары (входное изображение S^p $(p = \overline{1, L})$, необходимый выходной вектор t^p) выполняются шаги 4 – 6.

Шаг 4. Предъявляется выбранное входное изображение S^p, рассчитывается выходной вектор сети и определяется целевая функции $F_p(w_0)$, где W_0 – множество весов связей, полученных на предшествующем шаге алгоритма.

Шаг 5. Определяются приращения весов связей (в зависимости от разновидности алгоритма может меняться вес одной связи, веса связей групп или слоев нейронов, все веса нейронной сети) с помощью соотношения

$$\Delta w = \rho T(t) \mathrm{tg}(P(\Delta w)), \tag{6.39}$$

где Δw – случайное приращение выбранного веса связи; ρ – коэффициент скорости обучения; $T(t)$ – температура, определяемая соотношением (6.38); $P(\Delta w)$ – случайное число из интервала $(-\pi/2, \pi/2)$ с равномерным распределением.

Рассчитывается целевая функция с новыми значениями весов связей $F(w_0 + \Delta w)$.

Шаг 6. Если целевая функция улучшилась

$$(F(w_0 + \Delta w) < F(w_0)),$$

то сохраняются новые значения весов, если целевая функция ухудшилась $(F(w_0 + \Delta w) > F(w_0))$, то новые значения весов сохраняются с вероятностью, определяемой с помощью распределения Больцмана:

$$P(\Delta F) = e^{-\Delta F / kT}, \qquad (6.40)$$

где ΔF – приращение целевой функции; k – коэффициент, выбирается в зависимости от вида нейронной сети и решаемой задачи.

Для этого из интервала [0, 1] с равномерным распределением чисел выбирается случайное число q. Если $P(\Delta F) > q$, то изменение весов связей сохраняется, если $P(\Delta F) \le q$, то веса принимают предыдущие значения.

Шаг 7. Для предъявленной обучающей пары проверяется условие достижения заданного значения целевой функции. Если значение достигнуто и не выполняются условия прекращения работы алгоритма, то переход к шагу 3 и предъявление следующей обучающей пары, если значение целевой функции не достигнуто – то снижение температуры и переход к шагу 4 алгоритма.

Шаг 8. Останов.

Замечание 6.1. Приведенный алгоритм носит имя Коши, поскольку соотношение (6.39), определяющее случайное приращение веса связи, получено из распределения Коши:

$$P(x) = \frac{T(t)}{T(t)^2 + x^2},$$

где $P(x)$ – вероятность приращения величины x (при получении соотношения (6.39) полагали, что $x = \Delta w$).

За счет случайных шагов в направлениях, ухудшающих целевую функцию, алгоритм Коши позволяет вырываться из локальных минимумов, где небольшой шаг в любом направлении ведет к ухудшению целевой функции. Однако случайный поиск, выполняемый алгоритмом обучения Коши, имеет и существенный недостаток, так как он может требовать в сотни и тысячи раз больше времени, чем основной алгоритм обратного распространения.

Объединение методов обратного распространения и обучения Коши позволяет использовать достоинства каждого из них. Основной алгоритм обратного распространения дает преимущества прямого поиска – коррекция весов связей с его помощью всегда ведет к улучшению целевой функции и более быстрому по сравнению со случайным поиском определению минимумов целевой функции. Случайный поиск метода Коши, хотя и требует на порядки больше машинного времени, чем детерминированный поиск, однако позволяет определять глобальный минимум функции ошибки, поскольку в соответствии с выражением (6.39) возможны большие приращения весов связей, т.е. перемещение в многомерном и многоэкстремальном пространстве ошибки на значительные расстояния. Кроме того, при ухудшении целевой функции новые веса не отбрасываются автоматически, а сохраняются с некоторой вероятностью, определяемой с помощью распределения Больцмана (6.40). Это позволяет осуществлять поиск локальных минимумов во всех областях многомерного и многоэкстремального пространства ошибки, а информация о всех найденных локальных экстремумах и позволяет определить глобальный минимум целевой функции (или найти лучший набор весовых коэффициентов нейронной сети).

Контрольные вопросы

1. Какую область значений имеет функция активации нейрона?

а) (–1,1); б) (0,1); в) {0,1}; г) [0,1].

2. Точки {(4,–1), (8,–2), (1,1), (3,6)} принадлежат к классу A, а точки {(–8,4), (–2,–3), (–1,–1), (2,–9)} – классу B. Какой будет минимальная сеть, правильно классифицирующая эти точки:

а) нейрон с двумя входами;

б) нейрон с четырьмя входами;

в) однослойная сеть из двух нейронов с четырьмя входами;

г) двухслойная сеть с двумя входами, двумя нейронами в скрытом слое и одним нейроном в выходном слое.

3. Разработайте структуру сети Хебба, способную распознавать четыре различные буквы Вашего имени или фамилии. При этом обоснуйте выбор:

- числа рецепторных нейронов (число n x-элементов сети должно быть в пределах $12 \le n \le 30$);
- числа выходных нейронов;
- выбор векторов выходных сигналов.

2. Разработайте структуру элементарного перцептрона, способного распознавать первые две буквы Вашего имени и фамилии. При этом обоснуйте выбор:

✓ числа рецепторных нейронов (число n S-элементов перцептрона должно быть в пределах $12 \le n \le 30$) и числа нейронов скрытого слоя;

✓ величины шага в алгоритме обучения перцептрона;

✓ вида функций активации нейронов каждого слоя;

✓ величины порогов нейронов каждого слоя.

Смысла нет перед будущим дверь запирать,
Смысла нет между злом и добром выбирать.
Небо мечет вслепую игральные кости.
Все, что выпало, надо успеть проиграть.

Омар Хайям

7. МЕТОДЫ ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ

7.1. Эволюционное моделирование как метод решения проблем моделирования в условиях существенной априорной неопределенности

Моделирование процесса эволюции осуществлялось многократно, но только с появлением работы американских ученых Фогеля, Оуэнса, Уолша в науке об искусственном интеллекте появилось новое направление, заменяющее процесс моделирования интеллекта человека на моделирование процесса эволюции человеческого интеллекта [21].

В качестве объектов эволюции выбраны конечные автоматы, на входы которых поступают сигналы из внешней среды, представляющие собой источник последовательных сигналов из конечного алфавита.

На рис. 7.1 представлен конечный автомат *M* с тремя состояниями *A, B, C* алфавит входных символов автомата состоит из символов *a*, *b*, *c*, а выходных – из α, β, γ (входные символы приведены слева от наклонных черт, а выходные – справа, стрелкой отмечено начальное состояние автомата).

Рассмотрим работу автомата при поступлении на его вход последовательности символов *b*, *a*, *c*, *c*, a, *b*, *b*, *a*, *c*, *a*. Входной символ *b*, поданный на вход автомата в начальном состоянии *A*, приводит к появлению на выходе автомата

символа *a* и переходу автомата в состояние *B*. Следующий входной символ приходит на вход автомата только после того, как в автомате закончатся все переходные процессы. Второй символ *a* входной последовательности переводит автомат в состояние *C* с появлением на выходе автомата символа β. Покажем в табл. 7.1 последовательность изменения состояний автомата и выходных символов при заданной входной последовательности.

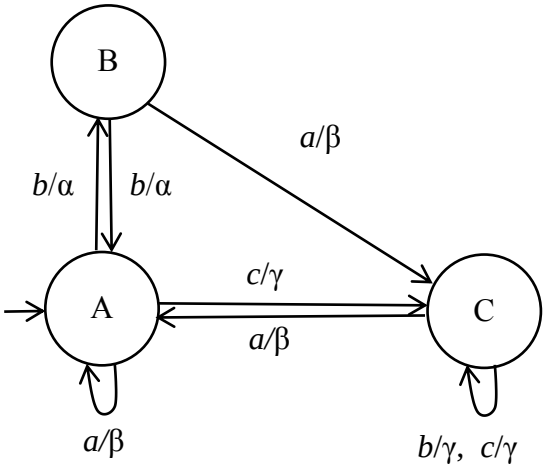


Рис. 7.1. Конечный автомат *M*

Таблица 7.1 – Функционирование конечного автомата

Текущее состояние	A	B	C	C	C	A	B	A	A	C
Входной символ	*b*	*a*	*c*	*c*	*a*	*b*	*b*	*a*	*c*	*a*
Следующее состояние	B	C	C	C	A	B	A	A	C	A
Выходной символ	α	β	γ	γ	β	α	α	β	γ	β

Отсюда видно, что автомат преобразует последовательность входных символов в последовательность выходных символов. При прогнозировании состояний внешней среди автомат должен иметь одинаковые входной и выходной алфавиты. Он может прогнозировать как следующий символ входной последовательности, поступающий на следующем такте работы автомата, так и любой другой символ с заданным числом тактов упреждения. Рассмотрим случай, ко-

гда автомат *M* прогнозирует следующий символ входной последовательности. При этом полагаем, что a ≡ α, b ≡ β, c ≡ γ, а стоимость ошибки равна единице при любом неправильном предсказании следующего входного символа и равна нулю при правильном предсказании (табл. 7.2). При функционировании автомата видно, что автомат правильно предсказал четыре входных символа и сделал пять ошибок.

Таблица 7.2 – Функционирование автомата

Текущее состояние	A	B	C	C	C	A	B	A	A	C	
Входной символ	*b*	*a*	*c*	*c*	*a*	*b*	*b*	*a*	*c*	*a*	
Прогнозируемый символ		*a*	*b*	*c*	*c*	*b*	*a*	*a*	*b*	*c*	*b*
Стоимость ошибки		0	1	0	1	0	1	0	1	1	

В эволюционном моделировании потомки автоматов получают путем случайных мутаций исходного автомата – добавляют или убирают одно или несколько состояний конечного автомата, изменяют начальное состояние автомата, изменяют конечное состояние у одного или нескольких ребер, входные и (или) выходные символы, соответствующие случайно выбранным ребрам автомата, и так далее. Полученные таким образом потомки оценивают на той же последовательности и, если они превосходят родительский автомат по заданному критерию, то один или несколько лучших автоматов оставляют для последующей работы, а остальные автоматы, включая и родительский, отбрасывают. Процесс мутаций повторяется до тех пор, пока не будет достигнуто определенное значение критерия качества функционирования автомата на заданной предыстории. Полученный в результате такого процесса автомат может использоваться для предсказания среды в реальном времени.

Синтез прогнозирующих автоматов средствами эволюционного моделирования можно рассматривать и как синтез предсказывающих фильтров, поиск

параметров которых ведутся методом случайного поиска. Известно, что эволюционирующие фильтры обычно имеют детерминированную часть, связанную с анализом причин и следствий, которую можно рассчитать, и индетерминированную часть – случайный корректор. Разумное сочетание детерминированных и индетерминированных вычислений позволяет существенно повысить эффективность алгоритмов эволюционного моделирования и расширить область их применения. В частности, это позволяет во многих случаях применять и более гибкий подход к эволюционному процессу, который дает возможность использовать конечные автоматы или непрерывные регуляторы в контурах управления объектов, находящихся в изменяющейся внешней среде. Он предполагает иерархическую двухуровневую процедуру, блок-схема которой приведена на рис. 7.2, где 1 – алгоритм структурной адаптации; 2 – алгоритм использования; 3 – алгоритм параметрической адаптации.

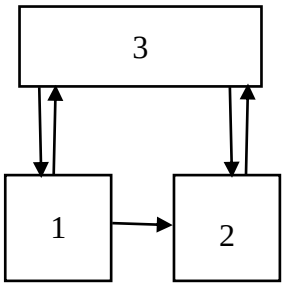


Рис. 7.2.Двухуровневая блок-схема
управления объектом

На первом уровне постоянно протекают два процесса: процесс использования одного из K хранящихся в памяти синтезированных регуляторов в контуре управления и процесс синтеза множества моделей регуляторов. При ухудшении качества работы функционирующего регулятора на i-м этапе он заменяется одним из синтезированных на $(i–1)$-м этапе регулятором либо одним из регуляторов, хранящихся в памяти. На $(i+1)$-м этапе функционирует новая модель регулятора и опять выполняется синтез множества моделей регуляторов. По-

скольку синтез эффективной структуры регулятора во многом зависит от таких параметров эволюционного процесса, как список возможных мутаций адаптируемых моделей, вероятности появления тех или иных режимов изменения, длины предыстории или объема используемой памяти на этапе адаптации и других параметров, то необходим алгоритм, который бы адаптировал параметры эволюционного процесса структурной адаптации моделей регуляторов. Такая адаптация выполняется с помощью алгоритма параметрической адаптации, который фактически адаптирует алгоритмы случайного поиска моделей регуляторов по поступающей из внешней среды информации и результатам функционирования регуляторов в контуре управления.

Адаптация случайного поиска существенно повышает эффективность алгоритмов эволюционного синтеза моделей, однако она не может полностью компенсировать отсутствие комплексного сочетания детерминированных и индетерминированных вычислений. Отсутствие такого сочетания приводит к тому, что там, где были бы эффективны простые детерминированные процедуры (например, модель находится на склонах “холма” или “пика” с глобальным экстремумом в многоэкстремальном пространстве возможных моделей и для получения лучшего решения необходимо только оптимизировать один-два параметра модели) применяется трудоемкий случайный поиск (при синтезе конечных автоматов генерируются автоматы во всем пространстве возможных решений, а не только на склонах “холма” или “пика”, на которых находится лучший синтезированный на данный момент автомат). Это часто чрезмерно увеличивает время счета, а следовательно, и ограничивает область применения эволюционного моделирования.

Для решения задач синтеза или адаптации автоматов в случае предельной априорной неопределенности лучше вводить универсальные алгебры $a(M, D)$, отражающие специфику решаемой задачи. Здесь M – множество возможных переходов $S_k^{i\alpha}S_l^{i\beta}$ конечных автоматов $B_i = \left\langle X^i, Y^i, S^i, \varphi_Y^i, \varphi_S^i \right\rangle$ из состояния $S_k^i \in S^i$

в состояние $S_l^i \in S^i$ при заданных конечных входном X^i и выходном Y^i алфавитах $(\alpha \in X^i, \beta \in Y^i)$; $i = \overline{1,n}$, – индекс, указывающий на принадлежность к *i*-му автомату; S^i – конечное множество состояний; $y_j^i = \varphi_Y^i\left(x_j^i, S_j^i\right)$ – функция выходов; $S_{j+1}^i = \varphi_S^i\left(x_j^i, S_j^i\right)$ – функция переходов; y_j^i, x_j^i и S_j^i – соответственно выходной символ, входной символ и состояние *i*-го конечного автомата B_i в момент времени t_j, j = 1, 2, ... ; D – множество операций алгебры, которая должна содержатьоперации, позволяющие наиболее просто решать задачу синтеза или адаптации автомата из некоторого множества $W = \left(B_1, B_2, ..., B_n\right)$ исходных автоматов при заданном множестве I исходных данных.

При заданной последовательности входных сигналов x_1^i, x_2^i, … , x_l^i выходные сигналы (реакции) y_j^i, $j = \overline{1,l}$, объектов B_i, $i = \overline{1,n}$, переводят наборы (I, B_1, B_2, …, B_n) в информационную матрицу $\left\|\beta_{kj}\right\|$, $k = \overline{1,n}$, $j = \overline{1,l}$, где $\beta_{kj} = P_j\left(B_k\right)$, $P = (P_1, P_2, ..., P_l)$ – множество предикатов, определенных на множестве автоматов, $P_q = P_q\left(B\right)$, $q = \overline{1,l}$. Строка $(\beta_{i1}, \beta_{i2}, ..., \beta_{il})$ информационной матрицы $\left\|\beta_{kj}\right\|_{k \times l}$ является информационным вектором автомата B_i и составлена из элементов 0 и 1, т. е. из значений предикатов, вычисленных при функционировании автомата. Если все элементы строки являются единицами, то автомат называется корректным для решаемой задачи Z. На множестве значений выходных функций вводятся метрики $\rho = (\rho_1, \rho_2, ..., \rho_\sigma)$, с помощью которых путем введения множества $\Phi = (\Phi_1, \Phi_2, ..., \Phi_r)$ функционалов на множестве значений выходных функций автоматов, оценивается отклонение функционирования некорректных автоматов от функционирования корректных. Если при функционировании автомата выполняются условия $\Phi_l \le \varepsilon_l$, $l = \overline{1,r}$, где $\varepsilon_l \ge 0$ – некоторые наперед заданные величины, то он называется автоматом с заданной мерой некорректности функционирования (при $\Phi_l = \varepsilon_l = 0$, $l = \overline{1,r}$ имеем корректный

автомат). При рациональном выборе множеств D, P, ρ, Φ обычно существуют признаки возможности или невозможности синтеза корректных автоматов или автоматов с заданной мерой некорректности функционирования из заданного множества W исходных объектов и множества I исходных данных. При наличии указанных признаков задача получения объекта для решения некоторой задачи Z выполняется следующим образом.

На первом этапе стандартным методом получается множество W^* автоматов, из которых по указанным признакам для данной задачи Z из множества W^* с помощью множества операций D алгебры $A(M, D)$ может быть получен корректный автомат (или автомат с заданной мерой некорректности). На втором этапе выполняется композиция или преобразование полученных объектов и для последующей работы отбирается подмножество лучших (в смысле заданного множества функционалов) автоматов, такое, что для данной задачи Z из них может быть получен корректный автомат (автомат с заданной мерой некорректности). Затем выполняется композиция или преобразование вновь полученных автоматов и так далее – до тех пор, пока не будет получен один или подмножество автоматов, каждый из которых решает заданную задачу Z. При решении задачи адаптации автомата в исходное множество W^* включается и ранее функционировавший автомат.

Успешный синтез конечного автомата, а также затраты на его осуществление как при использовании алгебраического подхода, так и классических алгоритмов эволюционного моделирования, во многом зависят и от удачного определения исходного множества автоматов B^*, из которых для данной задачи с помощью выбранного алгоритма можно синтезировать корректный (или с заданной мерой некорректности функционирования) автомат. В случае больших обучающих последовательностей и использования К-значных ($K >> 2$) входных и выходных алфавитов этап получения исходного множества автоматов может быть чрезвычайно трудоемким в связи с тем, что при его выполнении будет синтезировано множество бесперспективных частных описаний различной

сложности, которые затем будут опробованы на обучающих выборках и отброшены как работающие неудовлетворительно. В то же время простой предварительный анализ исходных данных может существенно сократить необходимые переборы частных описаний. Действительно, если, например, имеется обучающая последовательность, содержащая группу из j символов a_i и m пар символов $a_k a_u$ $(u = \overline{1,m})$,

$$a_1 a_2 a_3 \underbrace{a_i \ldots a_i}_{j-\text{раз}} a_k a_1 \ldots a_k a_2 \ldots a_k a_3 \ldots a_k a_m a_p \ldots,$$

и требуется синтезировать автомат, который безошибочно предсказывает все символы обучающей последовательности, то такой автомат должен иметь число состояний, которое определяется большим из двух чисел j и m, и, следовательно, на первом ряду селекции в множество B^* необходимо сразу вводить автоматы с $N = max(j, m)$ состояниями, не рассматривая более простые автоматы. Для определения числа m необходим просмотр всех пар следующих друг за другом входных символов, а для определения числа j – анализ групп входных символов, каждая из которых содержит j символов. В общем случае анализ входной последовательности можно представить следующим образом. Имеется входная последовательность $I = \{i_1, i_2, ..., i_n\}$ из алфавита $A = \{a_1, a_2, ..., a_m\}$ входных символов и имеется множество предикатов $P = \{P_1, P_2, ..., P_l\}$, которые характеризуют неоднородность участия символов алфавита A в образовании различных комбинаций символов во входной последовательности. Наличие или отсутствие интересующего нас свойства R_q входной последовательности будем описывать предикатом P_q, аргументами которого являются группы или отдельные символы входного алфавита A:

$$P_q(a_k, ..., a_r) = \begin{cases} 1, \text{если } \{i_1, i_2, ..., i_n\} \text{ обладает свойством } R_q, \\ 0 - \text{в противном случае.} \end{cases}$$

Рассмотрим, например, неоднородность участия в образовании входной

последовательности символов a_k и a_r. Искомую неоднородность можно характеризовать числами n_k, n_r вхождения каждого из символов во входную последовательность I и представить в виде диагональной матрицы F_1, в которой диагональные элементы $a_{ii} = n_i$, $i = \overline{1,m}$. Максимальное число $n_{i\max} = \max\limits_{i\in 1,m} a_u$ на диагонали матрицы F_1 дает грубую оценку максимально необходимого числа состояний корректного конечного автомата, безошибочно предсказывающего по текущему символу следующий символ входной последовательности.

Диагональные элементы матрицы F_1 можно использовать и для грубой оценки числа n_c состояний автоматов с заданной мерой некорректности. Например, если задано, что число ошибок предсказания автомата не должно превышать N_3, то используя соотношение

$$N_3 = \sum_{i=1}^{m} \varphi_i (n_i - n_c), \tag{7.1}$$

где

$$\varphi_i (n_i - n_c) = \begin{cases} n_i - n_c, \text{если } n_i \geq n_c, \\ 0, \text{если } n_i < n_c, \end{cases}$$

можно получить примерное значение числа n_c.

Видимо, оценки числа состояний корректных автоматов или автоматов с заданной мерой некорректности будут более точными, если наряду с числами n_i, $i = \overline{1,m}$, определять число пар входных символов, содержащих один из символов a_k или a_r, а также число упорядоченных пар $a_k a_r$ или $a_r a_k$. В общем случае, когда индексы k и r изменяются от единицы до m, можно построить частотную матрицу F_2 вида

$$F_2 = \begin{vmatrix} n_{11} & n_{12} & \dots & n_{1m} \\ n_{21} & n_{22} & \dots & n_{2m} \\ \dots\dots\dots\dots\dots\dots\dots \\ n_{m1} & n_{m2} & \dots & n_{mm} \end{vmatrix}, \qquad (7.2)$$

где n_{ij} – число упорядоченных пар $a_i a_j$ символов во входной последовательности.

Вместо матрицы (7.2) иногда удобнее рассматривать матрицу F_{2p} из значений предикатов

$$F_{2p} = \begin{vmatrix} P_{11} & P_{12} & \dots & P_{1m} \\ P_{21} & P_{22} & \dots & P_{2m} \\ \dots\dots\dots\dots\dots\dots\dots \\ P_{m1} & P_{m2} & \dots & P_{mm} \end{vmatrix}, \qquad (7.3)$$

где

$$P_{ij}(n_{ij}) = \begin{cases} 1, \text{если } n_{ij} \neq 0, \\ 0, \text{если } n_{ij} = 0, \ i, \ j = \overline{1,m}. \end{cases} \qquad (7.4)$$

Наибольшее число $n_{\max}$ отличных от нуля элементов в строке F_{2pi} матрицы (7.3) или в соответствующей строке матрицы F_{2i} (7.2) определяет нижнюю границу числа состояний конечного автомата, необходимых для правильного предсказания всех пар символов, в том числе и всех пар $a_i a_p$ ($p \in \overline{1,m}$) с наиболее разнообразно входящим в последовательность I символом a_i. Минимально необходимое для правильного предсказания всех символов число состояний автомата зависит не только от N_{max}, но и от величины максимального элемента $n_{\max} = \max\limits_{q,j} n_{qj}$ матрицы F_2, поскольку в худших случаях (например, все пары элементов $a_q a_j$ следуют одна за другой) для безошибочного предсказания необходимо $n_{\max}$ состояний конечного автомата. Однако в других случаях,

например, при циклической входной последовательности, когда группы пар символов $a_q a_j$ разделены одинаковыми подпоследовательностями символов, величина максимального элемента матрицы может существенно превышать минимально необходимое число состояний конечного автомата, требуемое для безошибочного воспроизведения входной последовательности. Такая неопределенность в оценке необходимого числа состояний конечного автомата требует дальнейшего дополнения характеристик входной последовательности и рассмотрения неоднородности участия троек, четверок, ..., *L*-ек символов в образовании свойств входной последовательности. Рассмотрим *L*-мерную матрицу

$$F_L = F\left(n_{i1,\,i2,\,\ldots,\,iL}\right),\ i_1, i_2, \ldots, i_L = \overline{1, m}.$$

Места на каждой стороне матрицы пронумеруем числами от единицы до *m*. Каждому числу $j = \overline{1, m}$ поставим в соответствие символ a_j алфавита *A*, а каждому элементу $n_{i1,\,i2,\,\ldots,\,iL}$ – число упорядоченных *L*-ек символов, в которые входят символы алфавита, соответствующие номерам индексов i_1, $i_{2,\,\ldots\,,}\ i_L \in \overline{1, m}$ выбранного элемента $n_{i1,\,i2,\,\ldots,\,iL}$ матрицы. Построенную таким образом матрицу называют *L*-мерной частотной матрицей входной последовательности.

Наличие *L*-мерной частотной матрицы F_L позволяет уточнить нижнюю оценку необходимого числа состояний конечных автоматов. Например, в случае, когда синтезируется автомат, который должен точно предсказывать все символы обучающей последовательности, наличие хотя бы одного отличного от нуля элемента $n_{i1,\,i2,\,\ldots,\,iL}$, когда $i_1 = i_2 = \ldots = i_L$ и $i_1 \in \overline{1, m}$, указывает, что минимально необходимое число состояний конечного автомата в общем случае не меньше *L*.

Вместо *L*-мерных матриц можно использовать более наглядные двумерные матрицы, пронумеровав их столбцы и строки таким образом, чтобы в них в удобной форме были перечислены все элементы *L*-мерной матрицы.

Отметим, что по аналогии с матрицей (7.3) вводятся многомерные или соответствующие им двумерные матрицы предикатов для L-ек символов.

Частотные матрицы целесообразно использовать и при синтезе конечных автоматов с заданной мерой некорректности функционирования.

Пример 7.1. Пусть задана последовательность из 22 символов $a_1, a_2, a_2, \alpha_2, a_3, a_1, a_1, a_4, a_2, a_2, a_2, a_4, a_1, a_4, a_5, a_2, a_2, a_2, a_1, a_3, a_1, a_5$. Необходимо синтезировать конечный автомат, точно предсказывающий по текущему символу следующий символ входной последовательности.

Матрицы F_1, F_2 в этом случае имеют вид

$$F_1 = \begin{vmatrix} 6 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{vmatrix}, \qquad F_2 = \begin{vmatrix} 1 & 1 & 1 & 2 & 1 \\ 1 & 6 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{vmatrix}.$$

Максимальный элемент матрицы F_1 указывает на то, что число состояний конечного автомата не превышает девяти, а из пяти ненулевых элементов первой строки матрицы F_2 следует, что автомат, безошибочно предсказывающий всю входную последовательность, не может иметь менее пяти состояний. Из величины максимального элемента второй строки матрицы F_2 и общего числа символов a_2 (максимальный элемент матрицы F_1) без анализа троек символов не удается снизить верхнюю границу числа состояний конечного автомата. Анализ троек символов с помощью матрицы

$$F_3 = \begin{array}{c|cccccccccccc} & a_1a_1 & a_1a_2 & \alpha_1a_3 & a_1a_4 & a_1a_5 & a_2a_1 & a_2a_2 & \dots & a_3a_1 & \dots & a_4a_1 & \dots \\ a_1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \dots & 1 & \dots & 0 & \dots \\ a_2 & 0 & 0 & 1 & 0 & 0 & 1 & 3 & \dots & 1 & \dots & 1 & \dots \\ a_3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & \dots & 0 & \dots \\ a_4 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \dots & 0 & \dots & 0 & \dots \\ a_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 & \dots & 0 & \dots \end{array}$$

показывает, что имеется три тройки символов a_2, a_2, a_2, отделенные друг от друга различными группами символов (все тройки символов a_1, a_2, a_k, (k = 1, 3, 4) различны). Поэтому число состояний автомата, безошибочно предсказывающего всю входную последовательность, не может быть меньше

$$n = k_1 n^*,$$

где k_1 – число групп символов a_2, a_2, a_2 (максимальный элемент матрицы F_3); n^* – число состояний конечного автомата, необходимых для безошибочного предсказания последовательности a_2, a_2, a_2, a_k. Отсюда следует, что n = 9 и что автомат должен состоять из трех подавтоматов, правильно предсказывающих соответственно подпоследовательности символов a_2, a_2, a_2, a_1; a_2, a_2, a_2, a_3; a_2, a_2, a_2, a_4. Такая информация на несколько порядков сокращает объем необходимых вычислений.

Пример 7.2. Пусть задана входная последовательность примера 1 и требуется оценить число N ошибочных предсказаний автомата с пятью состояниями.

Из соотношения (7.1) и матрицы F_1 примера 1 следует, что

$$N = \sum_{i=1}^{5} \varphi_1(n_i - 5) = 1 + 4 = 5.$$

Анализ матрицы F_2 показывает, что автомат с пятью состояниями, предсказывая вторые элементы пар символов $a_1 a_k$ ($k = \overline{1,5}$) может не допустить ни одной ошибки, т. е. верхняя оценка числа неправильных предсказаний, как будто, может быть уменьшена на единицу. Однако более глубокий анализ – анализ матрицы F_3 показывает, что тройки символов a_1, a_4, a_2 и a_1, a_4, a_5 можно пра-

вильно предсказать только в том случае, когда символ a_4 в них предсказывается разными состояниями автомата. Это свидетельствует о том, что величина оценки по первой строке матрицы F_1 числа ошибок с помощью соотношения (7.1) не может быть уменьшена.

Совместный анализ матриц F_2 и F_3, проведенный в примере 1, показывает, что девять состояний автомата, безошибочно предсказывающего все символы входной последовательности, обусловливаются наличием трех групп символов a_2, a_2, a_2, a_k (k = 1, 3, 4), для правильного предсказания каждой из которых требуется три состояния конечного автомата. Следовательно, автомат с пятью состояниями на подпоследовательностях, начинающихся с символа a_2, должен делать не менее двух ошибок. Поскольку величина оценки числа ошибок по третьей – пятой строкам матрицы F_1 с помощью соотношения (7.1) имеет нулевое значение, то весьма вероятно существование автомата с пятью состояниями, делающего всего три неправильных предсказания на заданной входной последовательности. Автомат, имеющий пять состояний и делающий всего три неправильных предсказания, действительно удается синтезировать.

Частотный метод анализа необходимо дополнять хотя бы простейшими элементами анализа во временной области. Дело в том, что важно не только наличие, например L-ки $a_k, \ldots, a_k$ символов, но и где эта L-ка расположена во входной последовательности: если в начале или в середине входной последовательности, то для безошибочного предсказания всех символов a_k и следующего за ними символа необходимо L состояний конечного автомата, а если в конце последовательности, то безошибочное предсказание может быть выполнено одним состоянием конечного автомата.

Другой факт из пространственно-временного анализа, позволяющий часто во много раз сократить объем необходимых вычислений. Если входная последовательность имеет длину N символов и выполняется синтез предсказывающего автомата, то теоретически нет смысла рассматривать конечные автоматы

с более чем $(N-1)$ переходами, а практически – более чем с $(0{,}5-0{,}7)(N-1)$ переходами.

Следовательно, при алгебраической форме представления автоматов необходимо оценивать только автоматы, содержащие не более $(0{,}5-0{,}7)(N-1)$ одночленов, что ведет к резкому сокращению числа возможных автоматов-претендентов.

7.2. Метод группового учета аргументов

Математическое моделирование различных природных и технических объектов и процессов – одно из важнейших направлений применения современной вычислительной техники искусственного интеллекта. Моделирование необходимо как для определения структуры, параметров и функций изучаемого объекта, так и для решения задач управления, прогнозирования и распознавания. Как показывает практика, моделирование реальных объектов сопряжено с рядом трудностей, связанных с плохой формализуемостью таких объектов, наличием размытых зависимостей, большого числа неопределенных параметров в аналитических моделях, недостаточной изученностью некоторых закономерностей, зашумленностью исходных данных и т.д. В таких случаях классические подходы построения моделей, основанные на высоком уровне знаний об объекте моделирования, как правило, неприемлемы, так как часто требуют на свое осуществление годы. Это обусловлено недостатками применяемой информационной технологии, предполагающей при получении математических моделей участие человека на каждом из ее трех основных этапов:

✓ установление на основе собранной информации основных причинно-следственных связей и механизмов, объясняющих изучаемый объект или явление, формулировка математической модели;

✓ анализ математической модели и разработка на ее основе алгоритмов решения проблемных задач;

✓ верификация модели и ее усовершенствование.

Поскольку человек, осуществляя поиск приемлемых решений, до 90 % работ выполняет вручную, то это и затягивает сроки моделирования сложных объектов на годы. Однако и прямые переборы моделей с помощью вычислительных средств, как правило, также невозможны из-за чрезмерного объема необходимых вычислений.

Специфика синтеза математических моделей в условиях существенной априорной неопределенности требует разработки новых и совершенствования известных методов и подходов к получению работоспособных математических моделей в таких условиях. В связи с изложенным актуально дальнейшее развитие методов обработки информации в условиях существенной априорной неопределенности, основанных на идеях эволюции и самоорганизации математических моделей, предложенных А.Г. Ивахненко [22-23].

С помощью алгоритмов метода группового учета аргументов (или метода самоорганизации математических моделей) решается большое число разнообразных задач синтеза математических моделей. Рассмотрим несколько основных типов этих задач, где успешно применяются алгоритмы метода группового учета аргументов (МГУА). При этом первую задачу сформулируем не в классическом виде, а в терминах теории равновесных состояний.

Задача 7.1. Пусть векторная вещественная функция $\varphi(y)=\{\varphi^1(y), \ldots, \varphi^q(y)\}$, $y \in \Omega \in E_n$ задана своими приближенными значениями $\tilde{\varphi}_\rho$ в ограниченном числе точек y_ρ, $\rho = 1, 2, \ldots, n$, $y_\rho \in Y_0 \in \Omega$. Необходимо найти приближенное значение $\tilde{\varphi}(y_i)$ в любой точке $y_i \in \Omega$ в виде

$$\varphi^i(y) = \sum_{k=0}^{m} a_k^i g_k(y), \; i = 1, 2, \ldots, q, \qquad (7.5)$$

где Ω – область определения функций g_k, φ^i; E_n – n-мерное векторное пространство; g_k, $k = 0, 1, \ldots, m$ – заданные или выбранные функции; Y_0 – множество,

содержащее r исходных точек; $A_1=\{a_0^1, \ldots, a_m^1, a_0^2, \ldots, a_m^2, \ldots, a_0^q, \ldots, a_m^q\}$ – вектор искомых коэффициентов.

Решение $z = (A_1^*, y^*)$, $A_1^* \in A(Y_1)$, $y^* \in Y_2$, $Y_1, Y_2 \in Y_0 \in \Omega$, $Y_1 \cup Y_2 = Y_0$, $Y_1 \neq Y_2$ ищется в предположении, что функция $\varphi_1(A_1,y)$, характеризующая точность модели на множествах Y_1 и y^*, и функция $\varphi_2(A_1,y)$, характеризующая ошибку модели на всем множестве Y_0 исходных данных и фактически определяющая подмножество точек $y^* \in Y_0$, в которых модель $\varphi(y)$ имеет максимальную погрешность, удовлетворяют следующим уравнениям

$$\varphi_1(A_1^*, y^*) = \max_{A_1 \in A(Y_1)} \varphi_1(A_1, y), \tag{7.6}$$

$$\varphi_2(A_1^*, y^*) = \max_{y_0 \in Y_2} \varphi_2(A_1^*, y_0), \tag{7.7}$$

где A – область допустимых значений вектора коэффициентов; Y_1 – множество точек исходных данных, на которых определяется вектор A_1 коэффициентов модели; y_0 – множество точек, на которых определяется ошибка моделирования исходных данных; Y_2 – множество точек исходных данных, на которых может определяться погрешность модели, полученной на множестве Y_1.

В качестве примера функции φ_1 могут служить функции вида $\varphi_1 = C - K$, где C – положительная константа, K – заданный критерий, причем, всегда выполняется условие $C \geq K$. Приведем примеры возможных критериев.

Критерий регулярности [24]

$$\Delta^2(Y_2) = \frac{\sum_{i=1}^{N_{np}} (\varphi(y_i) - \tilde{\varphi}(y_i))^2}{\sum_{i=1}^{N_{np}} (\tilde{\varphi}(y_i))^2}, \tag{7.8}$$

где $\varphi(y_i)$ – выходная величина модели, коэффициенты которой получены на

множестве Y_1 точек обучающей последовательности, в точке $y_i \in Y_2$; Y_2 – множество точек проверочной последовательности, $Y_1 \cup Y_2 = Y_0$, $Y_1 \cap Y_2 = \varnothing$; N_{np} – число точек в множестве Y_2; $\tilde{\varphi}(y_i)$ – фактическое значение функции $\varphi(y)$ в точке $y_i \in Y_2$.

Среднеквадратичная ошибка, вычисленная на отдельной проверочной последовательности

$$\delta^2(Y_2) = \frac{1}{N_{np}} \sum_{i=1}^{N_{np}} (\varphi(y_i) - \tilde{\varphi}(y_i))^2 , \; y_i \in Y_2. \qquad (7.9)$$

В качестве функции $\varphi_1(A_1, y)$ могут служить и другие критерии, модифицированные, если нужно, с учетом наличия множества y_0.

Примеры функций $\varphi_2(A_1, y)$.

Среднеквадратичный критерий, вычисленный на точках множества Y_0

$$\delta^2(Y_0) = \frac{1}{r_1} \sum_{j=1}^{r_1} (\varphi(y_i) - \tilde{\varphi}(y_i))^2 , \qquad (7.10)$$

где r_1 – число точек множества Y_0, на которых оценивается модель, $r_1 \le r$; r – число точек множества Y_0.

Критерии

$$\delta_1 = \max_{y_j \in Y_0} \left| \varphi(y_j) - \tilde{\varphi}(y_j) \right|, \qquad (7.11)$$

$$\delta_2 = \max_{y_{j1}, \ldots, y_{jq} \in Y_0} \sum_{k = j1, \ldots, jq} \left| \varphi(y_k) - \tilde{\varphi}(y_k) \right|, \; q << r, \qquad (7.12)$$

$$\delta_3 = \max_{y_j \in Y_2} \left| \varphi(y_j) - \tilde{\varphi}(y_j) \right| \qquad (7.13)$$

и так далее [23]. В выражениях (7.11) – (7.13) операция максимума берется на множестве Y_2 или Y_0, поэтому эти выражения можно использовать в виде соотношения (7.11).

В зависимости от вида функций $g_k(y)$, $\varphi^i(y)$, $i = 1, 2, \ldots, q$ и величины q в выражении (7.9) может порождаться целый ряд конкретных задач, решаемых различными алгоритмами МГУА.

Задача 7.2. Определение лучшей математической модели в виде скалярной функции ($q = 1$)

$$\varphi(y) = \sum_{k=0}^{m} a_k g_k(y). \tag{7.14}$$

При $\varphi(y) \equiv y$, $g_k(y) = x_k$, $k = 1, 2, \ldots, m$, имеем случай синтеза функции $y = f(x_1, x_2, \ldots, x_m)$.

Задача 7.3. Определение лучшей математической модели в виде системы линейных алгебраических уравнений: $q > 1$, $g_k(y) = y_k$; $\varphi^i(y) = y_i$, $i, k = 1, 2, \ldots, q$

$$\begin{aligned} y_1 &= a_0^1 + a_1^1 y_1 + \ldots + a_q^1 y_q, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ y_q &= a_0^q + a_1^q y_1 + \ldots + a_q^q y_q. \end{aligned} \tag{7.15}$$

Задача 7.4. Определение лучшей математической модели в виде обыкновенного дифференциального уравнения с постоянными коэффициентами: $q = 1$; $g_0(y) = 1$, $g_k(y) = y^k(t)$, $k = 1, 2, \ldots, m$; $\varphi(y) = \frac{dy}{dt}$; a_k, $k = 0, 1, \ldots, m$ – константы,

$$\frac{dy}{dt} = a_0 + a_1 y^1 + a_2 y^2 + \ldots + a_m y^m. \tag{7.16}$$

Задача 7.5. Определение лучшей математической модели в виде системы обыкновенных дифференциальных уравнений с постоянными или переменными коэффициентами: $q > 1$, $g_0(y) = 1$, $g_k(y) = y_k(t)$, $k = 1, 2, \ldots, m$; $\varphi^i(y) = \frac{dy_i}{dt}$,

$$\begin{aligned} \frac{dy_1}{dt} &= a_0^1 + a_1^1 y_1 + \ldots + a_q^1 y_q, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{dy_q}{dt} &= a_0^q + a_1^q y_1 + \ldots + a_q^q y_q. \end{aligned} \qquad (7.17)$$

где a_k^i, i = 1, 2, …, q, $k = 0, 1, \ldots, q$, соответственно постоянные коэффициенты или функции времени.

Если в задаче 7.5 положить, что $\varphi^i(y) = \int y_i(t)dt$, то поиск лучшей математической модели будет вестись в классе систем интегральных уравнений. Нетрудно также из задачи 7.1 получить задачи поиска математических моделей среди смешанных систем уравнений, например, содержащих алгебраические и дифференциальные уравнения или алгебраические и интегральные и т.д.

Из формулировок задач 7.1 – 7.5 и выражений (7.5), (7.14) – (7.17) в силу того, что правые части указанных соотношений могут рассматриваться как скалярные или векторные функции (или комбинации таких функций), то основу как этих, так и многих других задач синтеза моделей, решаемых алгоритмами самоорганизации, составляют задачи приближения функций одной, двух и большего числа переменных.

Обработку данных итерационными алгоритмами МГУА поясним на примере задачи 7.2, когда полное описание объекта ищется в виде некоторой нелинейной функции m переменных $y = f(x_1, x_2, \ldots, x_m)$. В отличие от регрессионного анализа, предполагающего известным вид уравнения регрессии и определяю-

щего только неизвестные коэффициенты уравнения, исходя из достижения минимума среднеквадратичной ошибки на всех точках исходных данных, метод самоорганизации математических моделей определяет не только параметры, но и структуру полного описания объекта. При этом выполняется последовательный синтез и селекция множеств постепенно усложняющихся моделей или решений – характерная особенность итерационных или многорядных алгоритмов самоорганизации математических моделей (или алгоритмов МГУА).

Блок-схема обработки данных этим методом при синтезе математического описания в виде функции m переменных $y = f(x_1, x_2, \ldots, x_m)$ приведена на рис. 7.3. Из схемы обработки данных следует, что структура и параметры нелинейной функции $f(x_1, x_2, \ldots, x_m)$ определяются посредством ряда последовательных шагов или рядов селекции. При этом все исходные данные делятся на две части – обучающую и проверочную последовательности.

На первом ряду селекции на точках обучающей последовательности генерируются простейшие частные модели как функции пар входных независимых переменных x_i, $i = 1, 2, \ldots, m$, например, вида

$$f(x_j,x_k) = a_1 + a_2x_j + a_3x_k,\ j \neq k,\ j = 1, 2, \ldots, (m-1),\ k = 2, 3, \ldots, m, \qquad (7.18)$$

или

$$f(x_j, x_k) = a_1 + a_2x_j + a_3x_k + a_4x_jx_k,$$

или

$$f(x_j, x_k) = a_1 + a_2x_j + a_3 x_j^2 + a_4x_k + a_5 x_k^2 + a_6x_jx_k$$

и т.д.

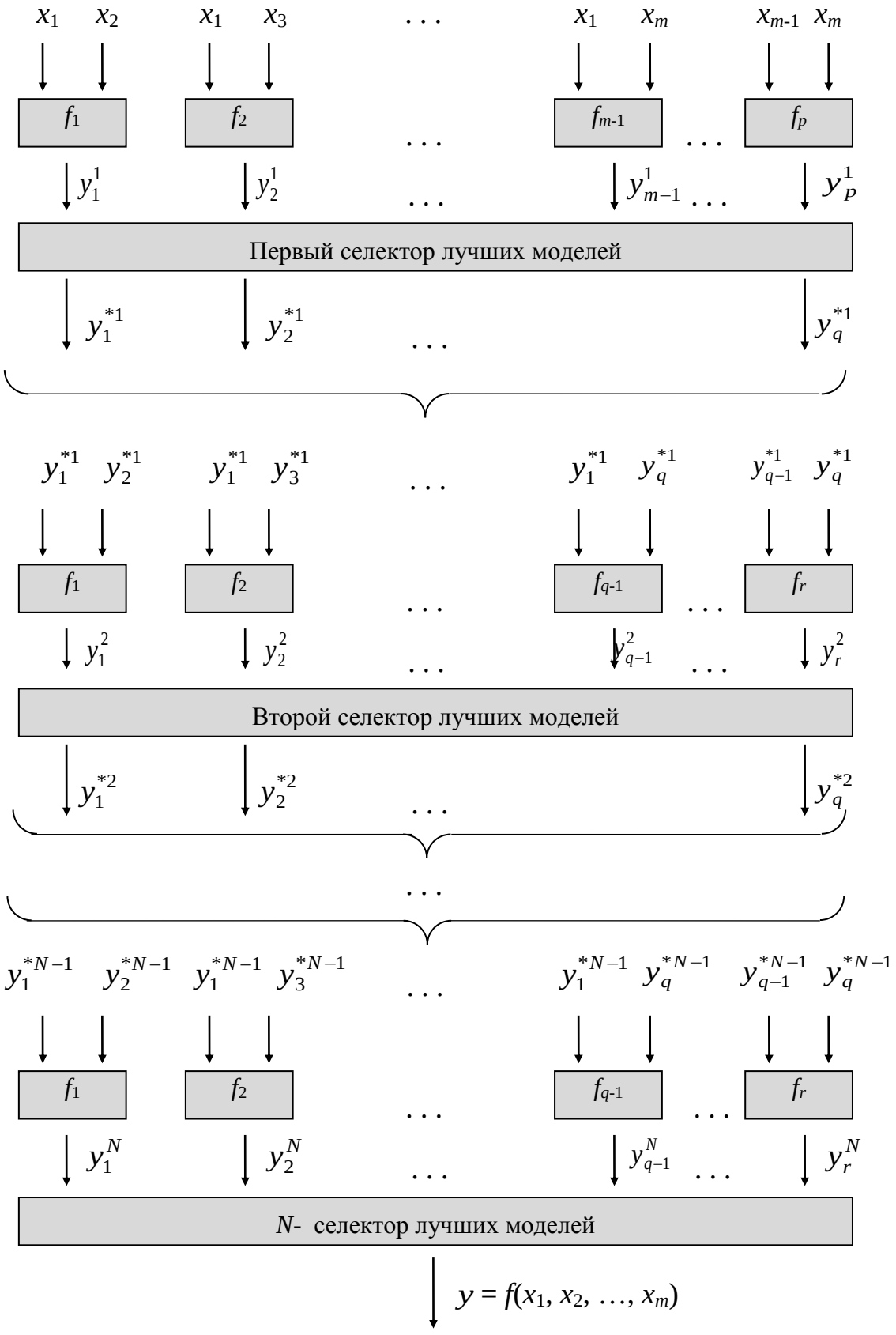


Рис. 7.3. Схема обработки данных многорядным алгоритмом МГУА

Все полученные модели первого ряда оцениваются на точках проверочной последовательности и q описаний, лучших по заданному критерию селекции, используются для синтеза моделей на точках обучающей последовательности на втором ряду селекции. Из полученного множества моделей на точках проверочной последовательности опять отбирается подмножество лучших, которые используются для синтеза моделей третьего ряда, и т.д. до тех пор, пока происходит улучшение показателя качества получаемых математических описаний.

Приведенная задача синтеза математического описания в виде скалярной функции m переменных является простейшим примером применения итерационных алгоритмов МГУА, которые используются для решения обширного класса задач моделирования.

В рассмотренном примере из ряда в ряд при помощи порогового отбора пропускаются только лучшие по принятому критерию селекции промежуточные описания. Как видно из рисунка и приведенных выражений, исходные аргументы в промежуточные переменные входят в соотношения попарно. Это справедливо для большинства алгоритмов МГУА, но известны и алгоритмы, в которых промежуточные модели получаются при использовании большего числа исходных аргументов или частных описаний предшествующих рядов селекции.

В зависимости от того, какая конкретная задача синтеза математической модели решается, какие используются критерии отбора моделей и соотношения для получения частных описаний, изменяется и общая схема алгоритма МГУА. Однако ряд блоков алгоритма остается при решении большинства задач автоматического синтеза математических моделей. Рассмотрим эти блоки на примере синтеза модели в виде нелинейно функции m переменных $y = f(x_1, x_2, \ldots, x_m)$ с помощью алгоритма МГУА с линейными полиномами. В этом алгоритме на первом и последующих рядах селекции используются соотношения вида (7.18). Алгоритм предполагает выполнение следующих основных шагов:

1. Задание основных параметров (числа и вида синтезируемых моделей на первом ряду селекции, числа лучших описаний, пропускаемых в следующий

ряд селекции, способа разделения исходных данных на части, критерия селекции, условий окончания работы алгоритма и т.д.).

2. Разделение исходных данных на обучающую и проверочную последовательности.

3. Получение нормальных уравнений Гаусса на точках обучающей последовательности.

4. Решение нормальных уравнений Гаусса.

5. Формирование массива частных описаний для текущего ряда селекции.

6. Оценка с помощью заданного критерия полученных частных описаний на множестве точек проверочной последовательности.

7. Отбор пропускаемых в следующий ряд селекции лучших частных описаний текущего ряда.

8. Проверка условий окончания процесса синтеза моделей. Если условия выполняются, то – переход на выполнение пункта 10.

9. Формирование частных описаний с линейными полиномами для следующего ряда селекции и переход на выполнение пункта 3.

10. Вывод информации о лучших полученных моделях.

11. Конец.

Все алгоритмы МГУА требуют разделения исходных данных на несколько частей. В наиболее простом и распространенном случае исходные данные делятся на обучающую и проверочную последовательности. Качество конечной модели во многом зависит от этого деления, однако универсального алгоритма, позволяющего наилучшим образом выполнить этот этап, нет. Наиболее часто применяются следующие способы деления исходных данных на две части:

– все исходные точки нумеруются числами натурального ряда, а затем обучающая последовательность формируется из точек с нечетными номерами, а проверочная – из точек с четными номерами, или наоборот;

– исходные точки ранжируются по дисперсии и в обучающую последовательность включается примерно половина точек с большей дисперсией, а в про-

верочную – оставшиеся точки;

– в проверочной последовательности используется только одна точка исходных данных;

– исходные данные ранжируются по дисперсии и делятся на обучающую и проверочную последовательности таким образом, чтобы получить минимум числа рядов селекции.

В лучших универсальных программах МГУА, как правило, используется целый спектр различных способов деления исходных данных на две и большее число частей.

В настоящее время известны сотни различных алгоритмов самоорганизации математических моделей, которые успешно применялись на универсальных вычислительных машинах фон-неймановской архитектуры при решении разнообразных задач математического моделирования. Однако известны и трудности применения алгоритмов самоорганизации, связанные как с недостаточной теоретической обоснованностью алгоритмов, так и нехваткой быстродействия самых лучших последовательных компьютеров. Дальнейшее повышение эффективности применения метода самоорганизации математических моделей связано не только с развитием теории метода, но и разработкой новых принципов организации аппаратных и программных средств, связанных, прежде всего, с распараллеливанием вычислений.

7.3. Основные понятия генетических алгоритмов

Генетические алгоритмы – это стремительно развивающееся направление в искусственном интеллекте. Под этим термином понимается большое число разнообразных алгоритмов, основанных на идеях генетики, изучающей законы и механизмы развития все живого на нашей планете. Эти алгоритмы, использующие принципы эволюционного отбора в природе, успешно применяются

для решения сложных задач поиска, оптимизации, синтеза математических моделей и обучения. Первые генетические алгоритмы были разработаны американским ученым Д. Холландом из Мичиганского университета для решения оптимизационных задач, требующих больших комбинаторных переборов [25].

В генетических алгоритмах информация об объектах или их математических моделях по аналогии с живыми организмами хранится в хромосомах (одной или нескольких), состоящих из генов. Каждый параметр объекта или его модели кодируется некоторым числом генов из фиксированного фрагмента определенной хромосомы. Фактически хромосома представляет собой вектор, компонентами которого являются параметры оптимизируемого объекта. В простейшем случае при двоично-десятичном кодировании параметров информация от родителей к потомкам передается с помощью трех генетических операторов – кроссовера, мутации и инверсии (рис. 7.4). По сравнению с оператором кроссовера последние два оператора играют второстепенную роль.

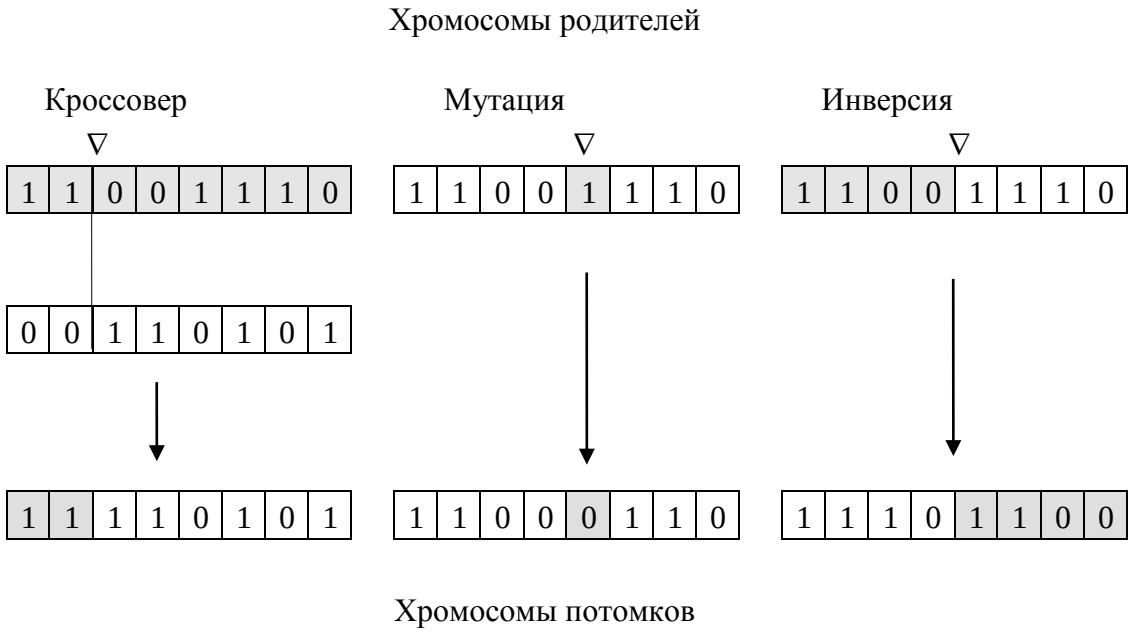


Рис. 7.4. Генетические операторы

При кроссовере потомок получает фрагменты хромосом двух родителей. Оператор мутации приводит к замене единицы на ноль и наоборот, а оператор инверсии – к изменению порядка следования фрагментов хромосомы. У всех трех операторов место приложения, помеченное значком ∇, выбирается случайным образом.

При решении с помощью генетических алгоритмов задачи синтеза математической модели некоторого объекта в условиях существенной априорной неопределенности на начальном этапе каким-либо образом генерируется множество исходных моделей или начальная популяция особей (блок 2 элементарного генетического алгоритма, рис. 7.5). Затем каждая сгенерированная модель (или особь популяции) оценивается с помощью заданного критерия (блок 3, рис. 7.5) и проверяется на достигнутое качество функционирования (блок 4). Как и в живой природе, в генетических алгоритмах преимущественное право для производства потомков с помощью простого копирования и (или) генетических операторов получают лучшие модели (или наиболее приспособленные к данной среде особи популяции), которые используются для генерирования нового множества моделей или особей популяции (блок 7 алгоритма) для ее следующего такта жизни. Отбор лучших особей может выполняться различными способами. Отметим среди них наиболее распространенные: пропорциональный отбор (или метод "рулетки"), ранжирование, случайный отбор, локальный отбор, отбор на основе усечения, турнирный отбор, метод Больцмана [26].

Все особи новой популяции также оцениваются с помощью критерия качества (блок 3) и лучшие из них получают преимущественное право для получения потомков следующего поколения. Этот процесс продолжается до тех пор, пока не будет достигнуто необходимое качество синтезируемых моделей (приспособленности особей популяции к данной среде) или не будут выполняться другие условия окончания вычислений (например, связанные с временными затратами или с вырождением популяции), которые анализируются в блоке 4 алгоритма.

Рис. 7.5. Элементарный генетический алгоритм

Высокая эффективность генетических алгоритмов при решении различных задач привела к разработке не только многочисленных программных продуктов, но и аппаратно-программных средств [27, 28], где появилась поддержка обучения не только на программном, но и на аппаратном уровне. При этом используются способные к эволюции аппаратные средства (СЭАС, Evolvable Hardware (EHW)), которые обеспечивают адаптацию своей архитектуры в соответствии со входными данными (или реакцией окружающей среды) или отказами оборудования. Реконфигурация оборудования дает возможность увеличить скорость адаптации в сотни и тысячи раз, что позволяет технической системе в реальном времени адекватно реагировать на поведение быстро изменя-

ющейся внешней среды или внезапно возникающие отказы внутри самой системы. СЭАС уже реализованы в виде отдельных микросхем. Архитектура БИС СЭАС приведена на рис. 7.6.

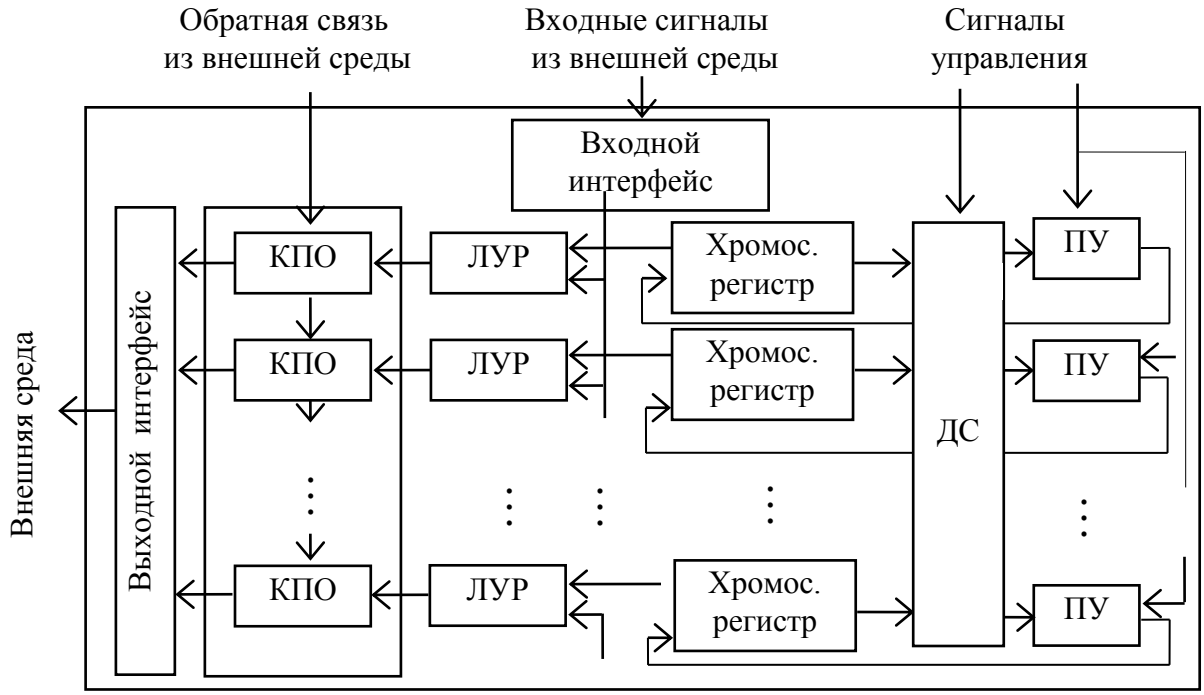


Рис. 7.6. Апаратно-программная реализация
генетических алгоритмов

В ее состав входят три типа компонент:

1. Логические устройства реконфигурации аппаратных средств (ЛУР). (ЛУР – это дальнейшее развитие программируемых логических матриц, в которых во много раз повышена как скорость записи, так и число возможных перезаписей данных.)

2. Параллельная аппаратура генетического алгоритма (вектор регистров хромосом, вектор процессорных устройств (ПУ) и дистрибутивная сеть (ДС)).

3. Компоненты поддерживающего обучения (КПО).

Архитектура функционирует следующим образом. Входные сигналы из окружающей среды вводятся через входной интерфейс в каждое ЛУР. Архитектура любого ЛУР определяется конфигурационными битами хромосомы из со-

ответствующего регистра. Следовательно, выход каждого ЛУР зависит как от входных сигналов из внешней среды, так и от хромосомы. Выходные сигналы со всех ЛУР поступают на входы компонент поддерживающего обучения. Сигналы с выходов КПО через выходной интерфейс воздействуют на внешнюю среду. Вычислительной системой производится оценка приспособленности каждой из хромосом к внешней среде. Затем на основе полученных оценок параллельная аппаратура генетического алгоритма по сигналам устройства управления системы выполняет над имеющимися хромосомами преобразования, соответствующие аппаратно реализованному генетическому алгоритму.

7.4. Метод рулетки

Каждой хромосоме может быть сопоставлен сектор колеса рулетки, величина которого устанавливается пропорционально значению функции приспособленности (ФП) данной хромосомы, т.е. чем больше значение ФП, тем больше сектор на колесе рулетки [25]. Все колесо рулетки соответствует сумме значений ФП всех хромосом рассматриваемой популяции.

Каждой хромосоме ch_i для $i = 1, 2, \ldots, K$ (K – численность популяции) соответствует сектор колеса $v(ch_i)$ выраженной в %

$$P_s = \frac{F(ch_i)}{\sum_{j=1}^{K} F(ch_j)} 100\%,$$

где $F(ch_i)$ – ФП i-й хромосомы; $p_s(ch_i)$ – вероятность селекции хромосомы ch_i.

Чем больше сектор, тем больше вероятность победы соответствующей хромосомы и соответственно в среднем функция приспособленности от поколения к поколению будет возрастать.

Пример 7.3. Пусть необходимо максимизировать функцию $f(x) = x^2 + 1$ на

отрезке [0; 30].

1. Проведем инициализацию популяции.

1.1. Решение задачи подвергаем бинарному кодированию – для представления решения (числа от 0 до 30) достаточно 5 битов, т.е. хромосома будет длиной 5 битов.

1.2. Выберем размер популяции $N = 8$. (Для реальных вычислений задач значение N значительно увеличивают).

Произвольно выбираем 8 точек на отрезке [0; 30], например:

$$7 \quad 5 \quad 14 \quad 21 \quad 26 \quad 19 \quad 8 \quad 5.$$

Переводим значения в двоичный вид:

$$Ch_1 = [00111], \quad Ch_2 = [00101], \quad Ch_3 = [01110], \quad Ch_4 = [10101],$$
$$Ch_5 = [11010], \quad Ch_6 = [10011], \quad Ch_7 = [01000], \quad Ch_8 = [00101].$$

Получили первоначальную популяцию из 8 особей.

2. Рассчитываем ФП каждой особи

$$F_1(ch_1) = ch_1^2 + 1 = 50, \quad F_2(ch_2) = ch_2^2 + 1 = 26,$$
$$F_3(ch_3) = ch_3^2 + 1 = 197, \quad F_4(ch_4) = ch_4^2 + 1 = 442,$$
$$F_5(ch_5) = ch_5^2 + 1 = 677, \quad F_6(ch_6) = ch_6^2 + 1 = 365,$$
$$F_7(ch_7) = 2 * ch_7^2 + 1 = 65, \quad F_8(ch_8) = 2 * ch_8^2 + 1 = 26.$$

3. Следующий шаг – селекция хромосом методом рулетки. На основании формулы

$$p_s = \frac{F(ch_i)}{\sum_{J=1}^{K} F(ch_J)} * 100\%,$$

получаем секторы колеса рулетки в % для каждой из восьми особей (рис. 7.7).

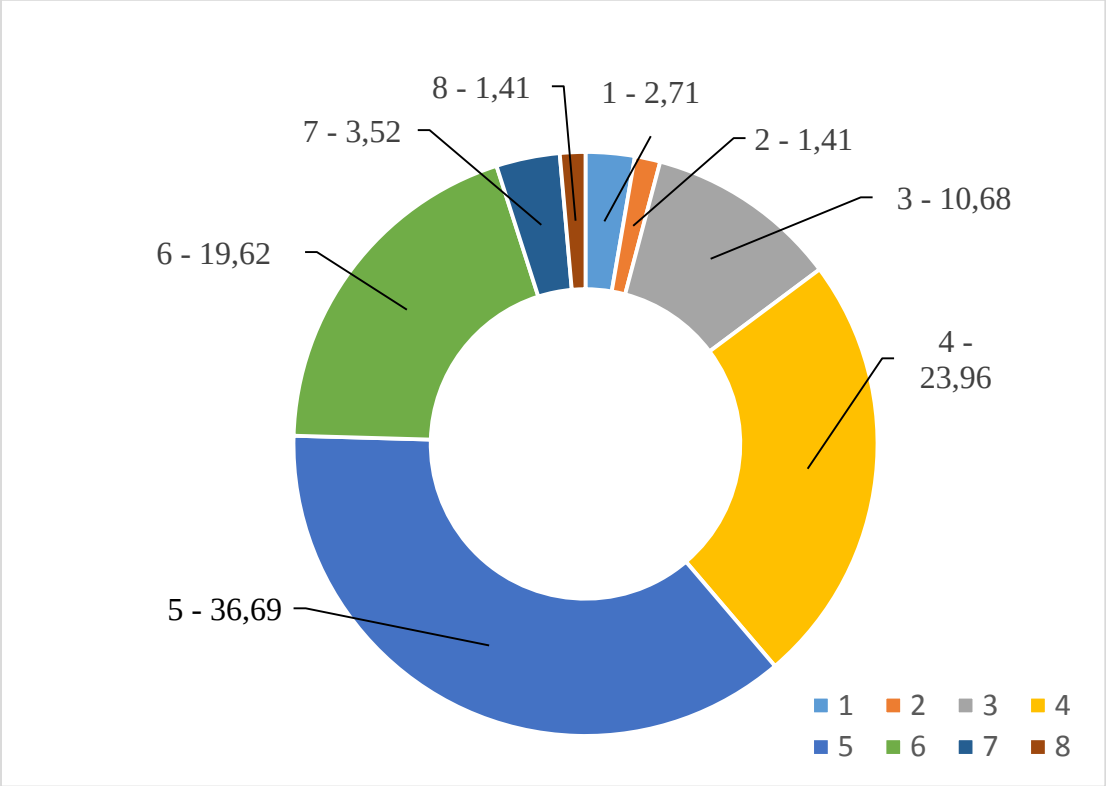


Рис. 7.7. Секторы колеса рулетки в %

4. Далее крутим рулетку – выпадают числа от 0 до 100. Пусть выпали следующие числа:

81 45 8 77 37 96 43 29.

Идентифицируем, в какой сектор попали эти числа, то есть какие хромосомы участвуют в скрещивании. Получаем следующий выбор хромосом – родительский пул (временную популяцию, нужную для формирования потомков):

6 5 3 6 4 7 5 4.

5. Допустим, что ни одна из хромосом не подвергается мутации $p_m = 0$.

6. Объединяем пары, по порядку нахождения их в родительском пуле.

Находим для каждой пары точки разреза хромосом t_k путем случайного выбора целых цифр из диапазона [1, 4] (табл. 7.3).

Таблица 7.3 – Скрещивание хромосом

Первая пара родителей Ch_6 = [10011] Ch_5 = [11010] t_k = 3	Первая пара потомков [10010] [11011]
Вторая пара родителей Ch_3 = [01110] Ch_6 = [10010] t_k = 2	Вторая пара потомков [01010] [10110]
Третья пара родителей Ch_4 = [10101] Ch_7 = [10010] t_k = 1	Третья пара потомков [11000] [00101]
Четвертая пара родителей Ch_5 = [11010] Ch_4 = [10101] t_k = 2	Четвертая пара потомков [11101] [10010]

Получили следующее, второе, поколение популяции. Этим хромосомам на отрезке [0, 30] соответствуют числа:

18 26 10 22 24 5 29 18.

Вычисляем F_i (ch_i) для каждой новой особи:

325 677 101 485 577 26 842 325.

Получили популяцию с особями с более высоким уровнем приспособленности (F_7 = 842) чем в исходной популяции.

Цикл повторяется до условия останова.

Достоинством генетических алгоритмов является:

1) Отсутствует ограничение на дифференцируемость функций. В частности, ГА работает и тогда, когда функции нет вообще.

2) Гибкость – хорошо работает при минимуме информации об окружающей среде (при высокой степени априорной неограниченности).

3) В ряде случаев ГА может находить только логический минимум (максимум). Несмотря на это, часто дает быстрое нахождение приемлемого решения.

4) Комбинируется с другими методами искусственного интеллекта, и его эффективность может повышаться.

5) Применяются для решения поисковых задач, которые имеют большое пространство поиска решений с целью уменьшения этого пространства поиска. Наиболее распространенное применение – решение задач оптимизации.

6) Минимизация ошибок.

7) Высокая скорость поиска решений.

8) Возможность распараллеливания вычислений.

9) Комплексная, а не поэтапная оптимизация задачи.

Контрольные вопросы

1. Что такое метод группового учета аргументов и для каких задач он применяется?

2. Какие виды опорных функций применяются для МГУА?

3. Какие методы разделения исходных данных на обучающую и проверочную последовательности существуют?

4. Какие две последовательности входных данных используются в МГУА?

5. Как влияет число аргументов на первом ряду селекции на точность прогнозирования?

6. Как влияет число пропускаемых в следующий ряд селекции частных описаний на точность прогнозирования?

7. Как влияет критерий отбора частных описаний на результаты прогнозирования?

8. Сколько частных описаний будет синтезировано на втором ряду селекции, в случае, если на второй ряд пропущено 11 лучших моделей?

9. С помощью каких методов могут определятся коэффициенты моделей на точках обучающей последовательности?

10. Что моделируют ГА?

11. Какие задачи решают с помощью ГА?

12. Что входит в понятие хромосома? Способы кодирования хромосом.

13. Что такое ген и особь?

14. Как работает метод рулетки?

15. Что такое генетический оператор кроссовер?

16. Что такое мутация, селекция? Какие есть их виды?

17. Что такое функция приспособленности?

18. Перечислите достоинства ГА.

Знание – сила.

Френсис Бэкон

8. ТЕХНОЛОГИИ DATA MINING

Data Mining – это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей [29].

Data Mining – это процесс обнаружения в сырых данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Суть и цель технологии Data Mining можно охарактеризовать так: это технология, которая предназначена для поиска в больших объемах данных *неочевидных, объективных и полезных* на практике закономерностей.

Неочевидных – это значит, что найденные закономерности не обнаруживаются стандартными методами обработки информации или экспертным путем.

Объективных – это значит, что обнаруженные закономерности будут полностью соответствовать действительности, в отличие от экспертного мнения, которое всегда является субъективным.

Практически полезных – это значит, что выводы имеют конкретное значение, которому можно найти практическое применение.

Знания – совокупность сведений, которая образует целостное описание, соответствующее некоторому уровню осведомленности об описываемом вопросе, предмете, проблеме и т.д.

Data Mining – это процесс выделения из данных неявной и неструктурированной информации и представления ее в виде, пригодном для использования.

Data Mining – это процесс, цель которого – обнаружить новые значимые корреляции, образцы и тенденции в результате просеивания большого объема хранимых данных с использованием методик распознавания образцов плюс применение статистических и математических методов.

В основу технологии Data Mining положена *концепция шаблонов*, которые представляют собой *закономерности*, свойственные подвыборкам данных, которые могут быть выражены в форме, понятной человеку.

Цель поиска закономерностей – представление данных в виде, отражающем искомые процессы. Построение моделей прогнозирования также является целью поиска закономерностей.

Data Mining как часть рынка информационных технологий относится к "Business Intelligence". Это программные средства, функционирующие в рамках предприятия и обеспечивающие функции доступа и анализа информации, которая находится в хранилище данных, а также обеспечивающие принятие правильных и обоснованных управленческих решений. Эти системы известны под названием систем поддержки принятия решений (СППР).

Большинство методов и алгоритмов**,** используемых в технологии Data Mining – это известные математические алгоритмы и методы:

– искусственные нейронные сети, деревья решений, символьные правила, метод опорных векторов, байесовские сети, линейная регрессия, корреляционно-регрессионный анализ; методы кластерного анализа;

– методы поиска ассоциативных правил, метод ограниченного перебора, эволюционное программирование, в том числе МГУА, и генетические алгоритмы, разнообразные методы визуализации данных и другие методы.

Традиционный процесс Data Mining включает следующие этапы:

✓ анализ предметной области;

- ✓ постановка задачи;
- ✓ подготовка данных;
- ✓ построение моделей;
- ✓ проверка и оценка моделей;
- ✓ выбор модели;
- ✓ применение модели;
- ✓ коррекция и обновление модели.

8.1. Стандарты Data Mining

Существуют стандарты, описывающие организацию процесса Data Mining и разработку Data Mining-систем.

Стандартный межотраслевой процесс является наиболее популярной и распространенной методологией Data Mining. В соответствии с этим стандартом Data Mining является непрерывным процессом со многими циклами и обратными связями и включает фазы:

1. Осмысление бизнеса (задачи).
2. Осмысление данных.
3. Подготовка данных.
4. Моделирование.
5. Оценка результатов.
6. Внедрение.

К этому набору фаз иногда добавляют седьмой шаг – контроль.

8.2. Классификация стадий Data Mining

Технология Data Mining может состоять из двух или трех стадий.

Стадия 1. Выявление закономерностей (свободный поиск).

Стадия 2. Использование выявленных закономерностей для предсказания

неизвестных значений (прогностическое моделирование).

*Стадия 3.*Анализ исключений – стадия предназначена для выявления и объяснения аномалий, найденных в закономерностях.

8.2.1. Свободный поиск.

На стадии свободного поиска осуществляется исследование набора данных с целью поиска скрытых закономерностей. Предварительные гипотезы относительно вида закономерностей здесь не определяются.

Закономерность – существенная и постоянно повторяющаяся взаимосвязь, определяющая этапы и формы процесса становления, развития различных явлений или процессов.

Система Data Mining на этой стадии определяет шаблоны, для получения которых в системах OLAP, например, аналитику необходимо обдумывать и создавать множество запросов. Здесь же аналитик освобождается от такой работы – шаблоны ищет за него система. Особенно полезно применение данного подхода в сверхбольших базах данных, где уловить закономерность путем создания запросов достаточно сложно, для чего требуется перепробовать множество разнообразных вариантов.

Свободный поиск представлен такими действиями:

- выявление закономерностей условной логики;
- выявление закономерностей ассоциативной логики;
- выявление трендов и колебаний.

Допустим, имеется база данных кадрового агентства с данными о профессии, стаже, возрасте и желаемом уровне вознаграждения. В случае самостоятельного задания запросов аналитик может получить приблизительно такие результаты: средний желаемый уровень вознаграждения специалистов в возрасте от 25 до 35 лет равен 1200 у.е.

Система сама ищет закономерности, необходимо лишь задать целевую переменную. В результате поиска закономерностей система сформирует набор

логических правил *если ..., то ....*

Могут быть найдены, например, следующие закономерности:

Если возраст < 20 лет и желаемый уровень вознаграждения > 700 у.е., то в 75% случаев соискатель ищет работу программиста

или

Если возраст > 35 лет и желаемый уровень вознаграждения > 1200 у.е., то в 90% случаев соискатель ищет руководящую работу.

Целевой переменной в описанных правилах в этом случае выступает профессия. При задании другой целевой переменной, например, возраста, получаем такие правила:

Если соискатель ищет руководящую работу и его стаж > 15 лет, то возраст соискателя > 35 лет в 65 % случаев.

Описанные действия, в рамках стадии свободного поиска, выполняются при помощи:

- ✓ индукции правил условной логики (задачи классификации и кластеризации, описание в компактной форме близких или схожих групп объектов);
- ✓ индукции правил ассоциативной логики (задачи ассоциации и последовательности и извлекаемая при их помощи информация);
- ✓ определения трендов и колебаний (исходный этап задачи прогнозирования).

На стадии свободного поиска также должна осуществляться *валидация закономерностей,* то есть проверка их достоверности на части данных, которые не принимали участие в формировании закономерностей. Такой прием разделения данных на обучающее и проверочное множество часто используется в ме-

тодах нейронных сетей и деревьев решений.

При технологии *дистилляции шаблонов* один образец (шаблон) информации извлекается из исходных данных и преобразуется в некие формальные конструкции, вид которых зависит от используемого метода Data Mining.

8.2.2. Прогностическое моделирование.

Здесь обнаруженные закономерности используются непосредственно для прогнозирования. Прогностическое моделирование включает такие действия:

- ✓ предсказание неизвестных значений;
- ✓ прогнозирование развития процессов.

В процессе прогностического моделирования решаются задачи классификации и прогнозирования.

При решении задачи классификации результаты работы первой стадии (индукции правил) используются для отнесения нового объекта, с определенной уверенностью, к одному из известных, предопределенных классов на основании известных значений.

При решении задачи прогнозирования результаты первой стадии (определение тренда или колебаний) используются для предсказания неизвестных (пропущенных или же будущих) значений целевой переменной (переменных).

Продолжая рассмотренный пример первой стадии, можно сделать вывод.

Зная, что *соискатель ищет руководящую работу и его стаж > 15 лет*, на 65 % можно быть уверенным в том, что *возраст соискателя > 35 лет*.

Или же,

Если возраст соискателя > 35 лет и желаемый уровень вознаграждения > 1200 условных единиц, то на 90% можно быть уверенным в том, что соискатель ищет руководящую работу.

Методы этой группы: логические методы; методы визуализации; методы кросс-табуляции; методы, основанные на уравнениях.

Логические методы, или методы логической индукции, включают: нечеткие запросы и анализы; символьные правила; деревья решений; генетические алгоритмы.

Методы этой группы являются наиболее интерпретируемыми – они оформляют найденные закономерности в большинстве случаев в достаточно прозрачном виде с точки зрения пользователя.

Полученные правила могут включать непрерывные и дискретные переменные. Следует заметить, что деревья решений могут быть легко преобразованы в наборы символьных правил путем генерации одного правила по пути от корня дерева до его терминальной вершины.

Деревья решений и правила фактически являются разными способами решения одной задачи и отличаются лишь по своим возможностям. Кроме того, реализация правил осуществляется более медленными алгоритмами, чем индукция деревьев решений.

Методы кросс-табуляции – это агенты и байесовские сети доверия.

Методы на основе уравнений, это когда закономерности представляются в виде математических уравнений. Основные методы данной группы: статистические методы (корреляционно-регрессионный анализ, корреляция рядов динамики, выявление тенденций динамических рядов, гармонический анализ) и нейронные сети.

8.2.3. Сравнение свободного поиска и прогностического моделирования с точки зрения логики.

Свободный поиск раскрывает общие закономерности. Он по своей природе индуктивен. Закономерности, полученные на этой стадии, формируются от частного к общему. В результате получаем некоторое общее знание о некотором классе объектов на основании исследования отдельных представителей

этого класса.

Правило:

Если возраст соискателя < 20 лет и желаемый уровень вознаграждения > 700 у.е., то в 75% случаев соискатель ищет работу программиста.

На основании частного, т.е. информации о некоторых свойствах класса

возраст < 20 лет

и *желаемый уровень вознаграждения > 700 условных единиц,*

делаем вывод об общем, а именно: соискатели – программисты.

Прогностическое моделирование, напротив, дедуктивно. Закономерности, полученные на этой стадии, формируются от общего к частному и единичному. Здесь получаем новое знание о некотором объекте или же группе объектов на основании:

– знания класса, к которому принадлежат исследуемые объекты;

– знания общего правила, действующего в пределах данного класса объектов.

Зная, что *соискатель ищет руководящую работу и его стаж > 15 лет*, на 65 % можно быть уверенным в том, что *возраст соискателя > 35 лет*.

На основании некоторых общих правил, а именно: цель соискателя – *руководящая работа и его стаж > 15 лет*, делаем вывод о единичном –

возраст соискателя > 35 лет.

8.2.4. Анализ исключений.

Для выявления отклонений необходимо *определить норму*, которая рассчитывается на стадии свободного поиска.

Например, найдено правило

Если возраст > 35 лет и желаемый уровень вознаграждения > 1200 у.е., то в 90 % случаев соискатель ищет руководящую работу.

Возникает вопрос – к чему отнести оставшиеся 10 % случаев?

Здесь возможны два варианта. Первый из них – существует некоторое логическое объяснение, которое также может быть оформлено в виде правила. Второй вариант для оставшихся 10% – это ошибки исходных данных. В этом случае стадия анализа исключений может быть использована в качестве очистки данных.

8.3. Задачи Data Mining

Классификация. В результате решения задачи классификации обнаруживаются признаки, которые характеризуют группы объектов исследуемого набора данных – классы; по этим признакам новый объект можно отнести к тому или иному классу.

Методы решения – ближайшего соседа; *k*-ближайших соседей; байесовские сети; индукция деревьев решений; нейронные сети.

Кластеризация. Кластеризация является логическим продолжением идеи классификации. Особенность кластеризации заключается в том, что классы объектов изначально не предопределены. Результатом кластеризации является разбиение объектов на группы.

Ассоциация. В ходе решения задачи поиска ассоциативных правил отыскиваются закономерности между связанными событиями в наборе данных.

Отличие ассоциации от двух предыдущих задач: поиск закономерностей осуществляется не на основе свойств анализируемого объекта, а между несколькими событиями, которые происходят одновременно.

Наиболее известный алгоритм решения задачи поиска ассоциативных правил – алгоритм Apriori.

Последовательность, или последовательная ассоциация позволяет найти временные закономерности между транзакциями.

Задача последовательности подобна ассоциации, но ее целью является установление закономерностей не между одновременно наступающими событиями, а между событиями, связанными во времени (т.е. происходящими с некоторым определенным интервалом во времени). Другими словами, последовательность определяется высокой вероятностью цепочки связанных во времени событий.

Фактически ассоциация является частным случаем последовательности с временным лагом, равным нулю.

Эту задачу Data Mining также называют задачей нахождения последовательных шаблонов например – *после события X через определенное время произойдет событие Y.* Например, после покупки квартиры жильцы в 60 % случаев в течение двух недель приобретают холодильник, а в течение двух месяцев в 50 % случаев приобретается телевизор. Решение данной задачи широко применяется в маркетинге и менеджменте, например, при управлении циклом работы с клиентом.

Прогнозирование. В результате решения задачи прогнозирования на основе особенностей исторических данных оцениваются пропущенные или же будущие значения целевых численных показателей. Для решения таких задач широко применяются методы математической статистики, нейронные сети и др.

Определение отклонений или выбросов – обнаружение и анализ данных, наиболее отличающихся от общего множества данных, выявление так называемых нехарактерных шаблонов.

Оценивание сводится к предсказанию непрерывных значений признака.

Анализ связей – задача нахождения зависимостей в наборе данных.

Визуализация. Примерами средств визуализации, при помощи которых можно оценить качество модели, являются диаграмма рассеивания, таблица сопряженности, график изменения величины ошибки. Пример методов визуализации – представление данных в 2D или 3D измерениях.

Подведение итогов – задача, цель которой – описание конкретных групп объектов из анализируемого набора данных.

8.4. Введение в ассоциативные правила

Впервые задача поиска ассоциативных правил была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины.

Рыночная корзина – это набор товаров, приобретенных покупателем в рамках одной отдельно взятой транзакции.

Транзакция – это множество событий, которые произошли одновременно.

Каждая такая транзакция представляет собой набор товаров, купленных покупателем за один визит.

Полученные в результате анализа *шаблоны* включают перечень товаров и число транзакций, которые содержат данные наборы.

Транзакционная или операционная база данных представляет собой двумерную таблицу, которая состоит из номера транзакции (*TID*) и перечня покупок, приобретенных вовремя этой транзакции.

TID – уникальный идентификатор, определяющий каждую сделку или транзакцию. Пример транзакционной базы данных, состоящей из покупательских транзакций, приведен в табл. 8.1.

Таблица 8.1 –Транзакционная база данных

TID	Приобретенные покупки	Приобретенные покупки
100	Хлеб, молоко, печенье	*a, b, c*
200	Молоко, сметана	*b, d*
300	Молоко, хлеб, сметана, печенье	*b, a, d, c*
400	Колбаса, сметана	*e, d*
500	Хлеб, молоко, печенье, сметана	*a, b, c, d*
600	Конфеты	*f*

На основе имеющейся базы данных нужно найти закономерности между событиями, то есть покупками.

Допустим, имеется транзакционная база данных *D*. Присвоим значениям товаров переменные (табл. 8.1, столбец 3)

Хлеб = *a*, Молоко = *b*, Печенье = *c*,
Сметана = *d*, Колбаса = *e*, Конфеты = *f*.

Рассмотрим набор товаров (*Itemset*), включающий, например, {Хлеб, молоко, печенье}. Выразим этот набор с помощью переменных: *abc* = {*a*, *b*, *c*}.

Этот набор товаров встречается в нашей базе данных три раза, то есть поддержка этого набора товаров равна 3:

$$\sup(abc) = 3.$$

При минимальном уровне поддержки, равной трем, набор товаров *abc* является часто встречающимся шаблоном, что запишем в виде

$$\min\_sup = 3, \{\text{Хлеб, молоко, печенье}\}.$$

Поддержкой набора называют количество или процент транзакций, содержащих определенный набор данных.

Для данного набора товаров поддержка, выраженная в процентном отношении, равна 50 %.

$$\text{SUP}(abc) = (3/6) \times 100\ \% = 50\ \%.$$

Поддержку иногда также называют *обеспечением набора*.

Таким образом, набор представляет интерес, если его поддержка выше определенного пользователем минимального значения. Эти наборы называют часто встречающимися.

Ассоциативное правило имеет вид: *Из события a следует событие b*.

В результате анализа устанавливают закономерность следующего вида:

Если в транзакции встретился набор товаров (или набор элементов) A, то можно сделать вывод, что в этой же транзакции должен появиться

набор элементов B.

Установление таких закономерностей дает возможность находить очень простые и понятные правила, называемые *ассоциативными*.

Основными характеристиками ассоциативного правила являются *поддержка* и *достоверность* правила.

Рассмотрим правило. "*Из покупки молока следует покупка печенья"* для базы данных, которая была приведена выше. В дополнение к понятию поддержки набора существует понятие *поддержки правила*.

Правило имеет поддержку s, если *s* % транзакций из всего набора содержат одновременно наборы элементов *a* и *b* или, другими словами, содержат оба товара.

Молоко – это товар *a*, печенье – это товар *b*. Поддержка правила "*Из покупки молока следует покупка печенья"* равна 3, или 50 %.

Достоверность правила показывает, какова вероятность того, что из события *a* следует событие *b*.

Правило "*Из a следует b"* справедливо с достоверностью *c*, если *c* % транзакций из всего множества, содержащих набор элементов *a*, также содержат набор элементов *b*.

Число транзакций, содержащих молоко, равно четырем, число транзакций, содержащих печенье, равно трем, достоверность правила равна (3/4) × 100%, т.е. 75 %. Достоверность правила "*Из покупки молока следует покупка печенья*" равна 75 %, то есть 75 % транзакций, содержащих товар *a*, также содержат товар *b*.

При помощи использования алгоритмов поиска ассоциативных правил аналитик может получить все возможные правила вида "*Из a следует b"*, с различными значениями поддержки и достоверности. Однако в большинстве случаев количество правил необходимо ограничивать заранее установленными минимальными и максимальными значениями поддержки и достоверности.

8.5. Методы поиска ассоциативных правил

Первый алгоритм поиска ассоциативных правил AIS был разработан сотрудниками исследовательского центра IBM в 1993 году. С этой работы начался интерес к ассоциативным правилам; на середину 90-х годов прошлого века пришелся пик исследовательских работ в этой области, и с тех пор каждый год появляется несколько новых алгоритмов.

В алгоритме AIS кандидаты множества наборов генерируются и подсчитываются сразу во время сканирования базы данных.

Недостаток алгоритма – излишнее генерирование и подсчет слишком многих кандидатов, которые в результате не оказываются часто встречающимися.

Для улучшения работы алгоритма был предложен *алгоритм Apriori.*

Работа данного алгоритма состоит из нескольких этапов, каждый из которых состоит из следующих шагов:

• формирование кандидатов;

• подсчет кандидатов.

Формирование кандидатов – этап, на котором алгоритм, сканируя базу данных, создает множество *i*-элементных кандидатов (*i*-номер этапа). На этом этапе поддержка кандидатов не рассчитывается.

Подсчет кандидатов – этап, на котором вычисляется поддержка каждого *i*-элементного кандидата. Здесь же осуществляется отсечение кандидатов, поддержка которых меньше минимума, установленного пользователем.

Оставшиеся *i*-элементные наборы называем часто встречающимися.

8.6. Алгоритм Apriori

Рассмотрим работу *алгоритма Apriori на примере* базы данных *D*. Иллюстрация работы алгоритма приведена на рис. 8.1.

Минимальный уровень поддержки принят равным 3. На первом этапе

происходит *формирование одноэлементных кандидатов.* Далее алгоритм подсчитывает поддержку одноэлементных наборов. Наборы с уровнем поддержки меньше установленного, то есть 3, *отсекаются.*

В примере это наборы *e* и *f*, которые имеют поддержку, равную 1. Оставшиеся наборы товаров считаются часто встречающимися одноэлементными наборами товаров: это наборы *a, b, c, d*. Далее происходит *формирование двухэлементных кандидатов*, подсчет их поддержки и отсечение наборов с уровнем поддержки, меньшим 3. Оставшиеся двухэлементные наборы товаров, считающиеся часто встречающимися двухэлементными наборами *ab, ac, bd*, принимают участие в дальнейшей работе алгоритма.

Если смотреть на работу алгоритма прямолинейно, на последнем этапе алгоритм формирует трехэлементные наборы товаров: *abc, abd, bcd, acd*, подсчитывает их поддержку и отсекает наборы с уровнем поддержки, менее 3. Набор товаров *abc* может быть назван часто встречающимся.

Однако алгоритм Apriori уменьшает количество кандидатов, отсекая априори тех, которые заведомо не могут стать часто встречающимися, на основе информации об отсеченных кандидатах на предыдущих этапах работы алгоритма. Отсечение кандидатов происходит на основе предположения о том, что у часто встречающегося набора товаров все подмножества должны быть часто встречающимися. Если в наборе находится подмножество, которое на предыдущем этапе было определено как нечасто встречающееся, этот кандидат уже не включается в формирование и подсчет кандидатов. Так, наборы товаров *ad, bc, cd* были отброшены как нечасто встречающиеся, алгоритм не рассматривал наборы товаров *abd, bcd, acd*.

TID	Покупки
100	*a, b, c*
200	*b, d*
300	*a, b, d, c*
400	*e, d*
500	*a, b, c, d*
600	*f*

Формирование 1-элементных кандидатов

Item-set	Support
a	3
b	4
c	3
d	4
e	1
f	1

Часто встречающиеся 1-элементные наборы

Itemset	Support
a	3
b	4
c	3
d	4

Формирование 2-элементных кандидатов

Itemset
ab
ac
ad
bc
bd
cd

Подсчет 2-элементных кандидатов

Item-set	Support
ab	3
ac	3
ad	2
bc	3
bd	3
cd	2

Часто встречающиеся 2-элементные наборы

Itemset	Support
ab	3
ac	3
bd	3
bc	3

Формирование 3-элементных кандидатов

Itemset
abc
abd
bcd
acd

Подсчет 3-элементных кандидатов

Item-set	Support
abc	3
abd	2
bcd	2
acd	2

Часто встречающиеся 3-элементные наборы

Itemset	Support
abc	3

Формирование 3-х элементных кандидатов

Itemset
abc

Подсчет 3-х элементных кандидатов

Item-set	Support
abc	3

Часто встречающиеся наборы

Itemset	Support
abc	3

Рис. 8.1. Пример работы алгоритма Apriori

При рассмотрении этих наборов формирование трехэлементных кандидатов происходило бы по схеме, приведенной в верхней части. Поскольку алгоритм априори отбросил заведомо нечасто встречающиеся наборы, последний этап алгоритма сразу определил набор *abc* как единственный трехэлементный часто встречающийся набор. На основании полученных данных, например, группируют ассортимент товаров в супермаркетах.

Для реализации задач Data Mining можно использовать систему Deductor – аналитическую платформу для создания законченных прикладных решений в области анализа данных. Реализованные в Deductor технологии позволяют на базе единой архитектуры пройти все этапы построения аналитической системы: от консолидации данных до построения моделей и визуализации полученных результатов.

Система Deductor состоит из пяти частей: Warehouse – хранилище данных, консолидирующее информацию из разных источников; Studio – приложение, позволяющее пройти все этапы построения прикладного решения, рабочее место аналитика; Viewer – рабочее место конечного пользователя, одно из средств тиражирования знаний; Server – служба, обеспечивающая удаленную обработку данных; Client – клиент доступа к Deductor Server. Обеспечивает доступ к серверу из сторонних приложений и управление его работой.

Контрольные вопросы

1. Какова суть и задачи, решаемые с помощьюData Mining?
2. Каковы этапы Data Mining?
3. Какие есть стадии Data Mining?
4. Что происходит на стадии свободного поиска?
5. Какие действия включает прогностическое моделирование?
6. Что происходит на стадии анализа исключений?
7. Перечислите методы, используемые при решении задач?
8. Какую задачу решают при поиске ассоциативных правил?
9. Что такое транзакция?
10. Что такое поддержка набора?
11. Что такое достоверность правила?
12. Приведите методы поиска ассоциативных правил.
13. Перечислите шаги алгоритма Apriori.
14. Назовите области применения результатов алгоритма Apriori.
15. Приведите примеры применения алгоритма Apriori.
16. Какова структура аналитической платформы Deductor?

В жизни нет ничего сложного. Это мы сложны. Жизнь – простая штука, и в ней что проще, тем правильнее.

О. Уайльд

9. МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ

Разработка технологии *искусственных агентов*, создание *мультиагентных систем* (МАС) (multiagent system) и *виртуальных организаций* представляет собой одну из наиболее важных и многообещающих областей развития информационных и коммуникационных технологий, с интеграцией современных сетевых *WWW*-технологий, методов и средств искусственного интеллекта, включая большие базы данных/знаний, многоядерные процессоры и системы объектно-ориентированного проектирования [30-32].

Сегодня это подход находит применение в таких областях как распределенное решение сложных задач, совмещенное проектирование изделий, реинжиниринг бизнес-процессов и построение виртуальных предприятий, имитационное моделирование интегрированных производственных систем и электронная торговля, организация работы коллективов роботов и распределенная разработка компьютерных программ.

9.1. Свойства агентных систем

Под *интеллектуальным агентом* (ИА) в слабом смысле понимается программно или аппаратно реализованная система, которая обладает такими свойствами:

✓ автономность – способность ИА функционировать без вмешательства

человека и при этом осуществлять самоконтроль над своими действиями и внутренним состоянием;

✓ общественное поведение – способность функционировать в сообществе с другими агентами, обмениваясь с ними сообщениями с помощью некоторого общепонятного языка коммуникаций;

✓ реактивность – способность воспринимать состояние среды и своевременно реагировать на те изменения, которые в ней происходят;

✓ про-активность – способность агента брать на себя инициативу, т.е. способность генерировать цели и действовать рационально для их достижения, а не только реагировать на внешние события.

Сильное определение агента предполагает, что у агента дополнительно имеется ряд ментальных свойств, дополняющих перечисленные выше, это:

- знания – это постоянная часть знаний агента о себе, среде и других агентах, т.е. та часть, которая не изменяется в процессе его функционирования;
- убеждения – знания агента о среде, в частности, о других агентах; это те знания, которые могут изменяться во времени и становиться неверными, однако агент может не иметь об этом информации и продолжать оставаться в убеждении, что на них можно основывать свои выводы;
- желания – это состояния, ситуации, достижение которых по разным причинам является для агента желательным, однако они могут быть противоречивыми и потому агент не ожидает, что все они будут достигнуты;
- намерения – это то, что агент или обязан сделать в силу своих обязательств по отношению к другим агентам, или то, что вытекает из его желаний;
- цели – конкретное множество конечных и промежуточных состояний, достижение которых агент принял в качестве текущей стратегии поведения;
- обязательства по отношению к другим агентам – задачи, которые агент берет на себя по поручению других агентов в рамках кооперативных целей или целей отдельных агентов в рамках сотрудничества.

Решение задачи одним агентом на основе инженерии знаний представляет собой точку зрения классического ИИ, согласно которой агент (или интеллектуальная система), обладая глобальным видением проблемы, имеет все необходимые способности, знания и ресурсы для ее решения. Напротив, *в распределенном искусственном интеллекте* и, вообще, в области МАС предполагается, что *отдельный агент* может иметь лишь *частичное представление* об общей задаче и способен решить лишь некоторую ее подзадачу. Поэтому для решения сколько-нибудь сложной проблемы, как правило, требуется *взаимодействие агентов*, которое неотделимо от организации МАС. Этот социальный аспект решения задач – одна из фундаментальных характеристик концептуальной новизны передовых компьютерных технологий и искусственных (виртуальных) организаций, строящихся как МАС [30].

Классификацию агентов обычно делают по степени развития внутреннего представления внешнего мира и по способу поведения.

В организационном *моделировании и проектировании* МАС выделяются следующие базовые процессы:

1) *функциональный* анализ и моделирование, направленные на спецификацию функций организации во всех ее измерениях;

2) *структурный* анализ и моделирование – определение возможных форм организаций и ряда основных параметров структуры;

3) *эволюционное проектирование,* включающее *функционально-структурный синтез* и *структурно-функциональный* анализ;

4) задание *параметров конкретизации*, определяющих переход от некоторой исходной организационной структуры к конкретной организации, то есть эффективную реализацию МАС.

Функциональный анализ МАС предполагает определение главных функций, возложенных на агентов. При этом организация может рассматриваться как система ролей, где каждая роль определяет множество характеристик агента в организации. Функции организации могут рассматриваться в различных

измерениях, соответствующих точкам зрения наблюдателей.

Структурный анализ МАС направлен на упорядочение множества возможных взаимодействий между агентами путем выделения связывающих их абстрактных отношений и анализа их изменений во времени.

Эволюционное моделирование МАС означает порождение и трансформацию ее функционально-структурной организации. Элементарный шаг процесса эволюционного проектирования включает две неразрывно связанные фазы: а) *функционально-структурный синтез,* при котором на входе имеем набор функций, а на выходе – совокупность структур; б) *структурно-функциональный анализ,* который начинается с рассмотрения ранее полученных структур, а завершается определением последующих функций.

При изменении условий существования МАС, росте сложности и неопределенности внешней среды, ее жизнеспособность может быть обеспечена лишь путем модификации (чаще всего, расширения) совокупности реализуемых ею функций, что обычно влечет за собой изменение структуры МАС.

Общая методология восходящего эволюционного проектирования МАС может быть представлена рекурсивной цепочкой: среда – функции МАС – роли агентов – агенты отношения между агентами – базовые структуры МАС – модификации, т.е. возможность изменения любого из звеньев указанной цепочки. Так модификации среды (внешних условий) приводит к изменениям функций МАС, которые влекут за собой порождение новых и замену ненужных агентов. Все это сопровождается сменой отношений между агентами и эволюцией исходных структур в интересах самосохранения системы путем ее приспособления к изменениям среды. Иными словами, эволюция МАС совместно со средой (или с другими МАС) предполагает и сохранение индивидуальности, и взаимную адаптацию.

Архитектуру агентной системы можно разделить на три функциональные части: моделирование, синхронизация времени и контроля, и часть, осуществляющую расчеты и визуализацию.

Общая методика *восходящего проектирования* МАС включает следующие этапы.

1) Формулирование назначения (цели разработки) МАС.

2) Определение типа и основных свойств среды МАС.

3) Определение основных и вспомогательных функций агентов в МАС.

4) Уточнение состава агентов и распределение функций между агентами; выбор архитектур агентов.

5) Выделение базовых взаимосвязей (отношений) между агентами в МАС.

6) Определение возможных действий (операций) агентов.

7) Построение базовой архитектуры (функционально-структурной единицы), анализ ее возможных состояний (нормальное, критическое и пр.).

8) Анализ реальных текущих или предполагаемых изменений внешней среды (условий функционирования) или внутренних противоречий МАС.

9) Определение соответствующих изменений функций агентов и формулирование стратегии эволюции МАС.

10) Построение общей архитектуры МАС, инвариантной к рассмотренной области изменений среды (или самовоспроизведение функционально-структурной единицы МАС).

Роли характеризуют функции агентов независимо от их внутренней структуры. По ролевому признаку возможны следующие виды агентов.

1. *Агент-заказчик,* рассылающий заявки на выполнение некоторого задания другим агентам.

2. *Агент-координатор* (посредник), который принимает заказы и распределяет их между другими агентами.

3.*Агент-исполнитель.*

4**.** *Агент-субкоординатор*, который организует и контролирует работу вышеуказанных трех видов агентов и наделен правом экстренного вмешательства и перераспределения ресурсов в критических ситуациях.

Эти роли соответствуют минимальному набору функций, необходимых

для функционирования организации как целенаправленной системы, т.е. достижения целей в условиях адаптации к изменениям среды.

Понятие *роли* определяется с помощью трёх атрибутов: *ответственности, разрешения, и протоколов.*

Для формального определения МАС в русле восходящего подхода можно взять за основу понятие *системы*, которая выражается в виде [30]

$$S = (X, \Pi, \Omega),$$

где X – непустое множество, называемое носителем или основой системы; Π – множество предикатов; Ω – множество операций. Очевидно, что система может быть многоосновной, и в этом случае $X = (X_1, \ldots, X_n)$. Мультиагентная система обычно включает как множество A *агентов*, так и множество манипулируемых ими *объектов* O, что может быть записано в виде $X = A \times O$.

В случае, когда $X = A$, эволюционная мультиагентная система определяется шестеркой [30]:

$$MAS = (X, E, R, AC, P, ST, EV),$$

где $X = A = \{1, ..., n\}$ – множество неоднородных агентов; E – множество сред, в которых может функционировать данная МАС; R – семейство базовых отношений между агентами, причем это семейство отношений включает, по крайней мере, три типа отношений и может быть представлено разбиением

$$R = R_1 \cup R_2 \cup R_3,$$

где R_1 – множество горизонтальных (симметричных) отношений; R_2 – множество асимметричных отношений, направленных “сверху вниз”; R_3 – множество нечетких асимметричных отношений, направленных “снизу вверх”; AC – множество действий агентов; P – множество коммуникативных актов, образующих

протокол коммуникации в МАС; *ST* – множество состояний МАС; *EV* – множество эволюционных стратегий [30].

Основные направления в МАС – это разработка *распределенного искусственного интеллекта* и *искусственной жизни*.

Идеология *распределенного искусственного интеллекта или* распределенного решения задач предполагает разделение знаний и ресурсов между агентами, и распределение управления и полномочий. Как правило, применяется единый орган управления, который обеспечивает принятие решений в конфликтных ситуациях. При этом исходным объектом исследования является общая сложная проблема, для решения которой формируется группа агентов, строится общая концептуальная модель и вводятся глобальные критерии достижения цели.

Распределенное решение задач несколькими агентами разбивается на следующие этапы:

1) проводится декомпозиция исходной проблемы на отдельные задачи;

2) эти задачи распределяются между агентами-исполнителями;

3) каждый агент-исполнитель решает свою задачу, возможно также разделяя ее на подзадачи;

4) для получения общего результата производится композиция, интеграция частных результатов, соответствующих выделенным задачам.

Второе направление – *искусственная жизнь* – в большей степени связано с трактовкой интеллектуального поведения в контексте выживания, адаптации и самоорганизации в динамичной, враждебной среде. Это и локальное взаимодействие большого числа простых и необязательно интеллектуальных агентов, *коллективный интеллект* или *интеллект роя*. При этом опираются на следующие положения:

1) МАС есть популяция простых и зависимых друг от друга агентов;

2) каждый агент самостоятельно определяет свои реакции на события в локальной среде и взаимодействия с другими агентами;

3) связи между агентами являются горизонтальными, т.е. не существует

агента-супервизора, управляющего взаимодействием других агентов;

4) нет точных правил, чтобы определить глобальное поведение агентов;

5) поведение, свойства и структура на коллективном уровне порождаются только локальными взаимодействиями агентов.

9.2. Теория коллективного поведения

Основная цель теории коллективного поведения (ТКП) – создание сложных систем, способных самостоятельно решать задачи без предварительного программирования человеком в условиях недостатка информации и быстрых изменений в окружении системы. Предметом ТКП являются принципы и методы построения таких систем. Основные задачи ТКП:

1. Исследование соотношения децентрализованного и централизованного управления.

2. Исследование вопросов об однородности и неоднородности коллективного поведения.

3. Исследование соотношения сотрудничества и соперничества в коллективе агентов.

4. Исследование коллективных моделей реальности (коллективное знание).

5. Вопрос коллективного принятия решений (голосование).

6. Вопрос общения (информационного взаимодействия) коллективных агентов. Языки общения агентов.

Интеллектуальная система (агент) – это компьютерная система, которая размещена в некоторой среде (рис. 9.1) и которая способна действовать в этой среде автономно для достижения целей, которые поставлены перед ней ее разработчиком. Поведение агента – это последовательность его действий, направленных на достижение поставленной цели в зависимости от состояния среды.

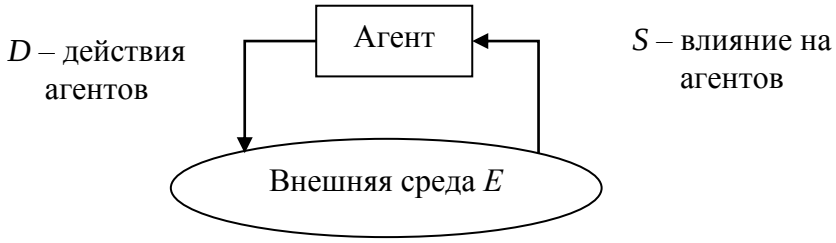


Рис. 9.1. Взаимодействие агента с внешней средой

Коллектив – это набор из *N* интеллектуальных систем (агентов), которые действуют вместе для достижения общих целей. Центр (пользователь) – тот, кто ставит задачи коллективу.

Функциональная целостность системы предполагает такой тип внутреннего взаимодействия ее элементов при котором свойства целого (т.е. всей системы) нельзя свести к сумме свойств элементов.

Принцип коллективного действия – гипотеза о преимуществе коллективного действия над индивидуальными (при определенных условиях) – коллективными действиями можно достичь больших результатов чем индивидуальными.

Решение задач человеком всегда сводится к выбору одного варианта из множества возможных. Почти каждую поведенческую задачу, которую решает человек, можно свести к формуле: укажите один элемент из некоторого множества. С опытом человек учится лучше решать задачу целенаправленного выбора.

Основная проблема, которую решает агент – это проблема выбора: какое из всех доступных действий выбрать и осуществить для того, чтобы достичь поставленную перед ним цель. Сложный поведенческий акт: это последовательность действий (последовательность элементарных выборов во времени).

Среду можно рассматривать как окружение агента, с которым он непосредственно взаимодействует. Сложность среды, в которой функционирует агент влияет на сложность процесса принятия решения.

С этой точки зрения среды можно разделить на типы по следующим классификационным признакам.

1. Доступные/недоступные. Доступная среда – это среда, в которой агент может получить полную, точную и последнюю информацию о состоянии среды.

Большинство сложных сред недоступны.

2. Детерминированные/недетерминированные. Детерминированная среда – это среда, в которой каждое действие имеет один и тот же гарантированный результат, то есть отсутствует неопределенность в отношении того, в какое состояние перейдет среда после реализации данного действия агента.

3. Эпизодические/неэпизодические. В эпизодической среде эффективность действия агента зависит от количества отдельных эпизодов взаимодействия со средой, которые не охвачены причинно-следственными связями. При этом отсутствует взаимосвязь между продуктивностью действий агента и различными сценариями развития событий. В более сложной эпизодической среде агент должен учитывать взаимосвязь между текущим и последующими эпизодами.

4. Статические/динамические. Статическая среда – это среда, которая не меняется, если агент не осуществляет в ней никаких действий. Динамичная среда – это среда, в которой протекают некоторые процессы, изменяющие состояние среды независимо от действий агента.

5. Дискретная/непрерывная. Среда называется дискретной если существует постоянное ограниченное число доступных в этой среде действий и характеристик среды, которые распознаются агентом.

На рис. 9.2 показана обобщенная структура агента, где *S* – сенсорная система, *E* – блок оценки, *D* – блок принятия решений, *A* – исполнительная система, *C* – блок информационного взаимодействия с другими агентами.

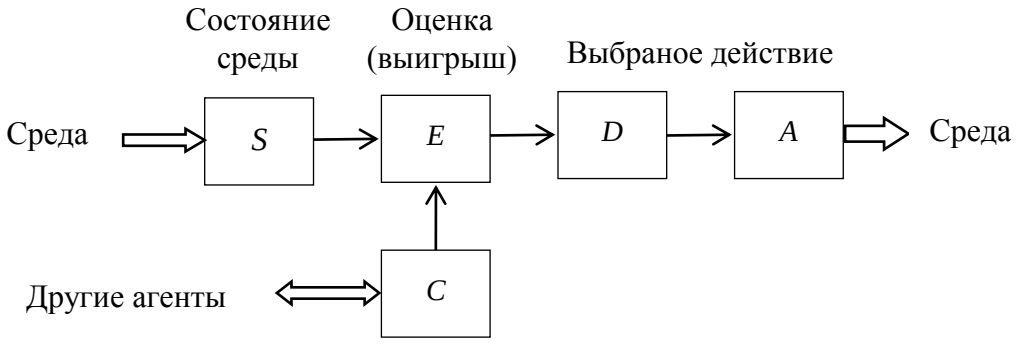


Рис. 9.2. Обобщенная структура агента

9.2.1. Формальное описание системы <среда, коллектив>.

Во всех следующих обозначениях верхний индекс задает идентификатор {состояния, действия}, а нижний индекс порядковый номер {состояния, действия} во времени.

Пусть $E = \{e^1, e^2, \ldots, e^k\}$ – множество всех возможных конечных состояний среды; $Ac = \{d^1, d^2, \ldots, d^n\}$ – набор доступных действий агента.

Тогда r – поведение (набор эпизодов взаимодействий агента со средой) можно представить, как операцию

$$r : e_0 \xrightarrow{d_0} e_1 \xrightarrow{d_1} e_2 \xrightarrow{d_2} \ldots \xrightarrow{dv-1} e_v.$$

Если R – множество всех возможных конечных последовательностей (поведений) для заданных E и Ac; $R^{Ac} \in R$ – все последовательности, которые заканчиваются действиями агента; $R^E \in R$ – все последовательности, которые заканчиваются состоянием среды.

Пусть $\tau : R^{Ac} \to f(E)$ – функция отображения (функция среды) отражает поведение агента R^{Ac} в некоторое подмножество состояний среды $Env = < E, e_0, \tau >$, т.е. среда – это тройка $< E, e_0, \tau >$, где E – множество всех возможных состояний среды, e_0 – начальное состояние среды, τ – функция внешней среды; $Ag : R^E \to A_c$ – функция агента, отображает предыдущую историю развития действий R^E в следующие действия агента, тогда коллектив, состоящий из N агентов можно представить

$$MAS = \{Ag_i\}, \; i = 1, \ldots, N.$$

Всю систему, включающую и среду и коллектив агентов опишем как

$$S =< Env, MAS >.$$

Для каждого текущего такта взаимодействия агента и среды рассчитывается функция оценки или выигрыша. Функцию оценки иногда называют откликом среды на действия агента.

Целевая функция рассчитывается на основе текущего значения функции выигрыша и значений функции выигрыша в предыдущих тактах взаимодействия. Функция оценки определяет каким способом агент накапливает свой интегральный выигрыш, на основе которого он принимает решение о выборе своих следующих действий.

Обычный способ задания целевой функции состоит в том, чтобы запрограммировать выполнение агентами последовательности действий при возникновении определенных обстоятельствах. Но этот способ не подходит, если возможны непредсказуемые или неизвестные наперед обстоятельства. В этом случае нужно сообщить агенту что делать, вместо – каким способом это делать.

Одним из возможных подходов такого решена есть непрямое описание задачи с помощью некоторой меры эффективности (задача оптимизации).

Покажем два основных способа определения функции оценки выигрыш как показателя эффективности действий агента.

1. Каждому состоянию среды (множество E) соответствует некоторое числовое значение (например, из множества действительных чисел P) показателя эффективности u^1. Это значение показывает насколько выгодным для агента есть это состояние среды

$$u^1 : E \to P.$$

2. Каждому из всех возможных поведений агента (множество R) соответствует некоторое значение показателя эффективности. Эта оценка является долговременной

$$u^2 : R \to P.$$

Целевая функция агента определяет каким образом показатели эффективности различных состояний среды (или при разном поведении агента) объединяются в одну величину, которая учитывается блоком принятия решений при выборе следующих действий агента

$$Ag : R^E \xrightarrow{\varphi(\mathrm{u})} Ac.$$

Как правило, перед агентом ставится задача максимизировать значение целевой функции φ(u). Вид целевой функции выбирается разработчиком в зависимости от задачи, для решения которой проектируется агент.

Ниже даны примеры целевых функций, где T – текущее количество тактов взаимодействия со средой, u_j – показатель эффективности действий агента (выигрыш) на j-м такте взаимодействия со средой:

1. Агент заинтересован в нахождении состояния среды с наибольшим u_j

$$\varphi(u_j) = \max_{e_i \in E} \{u_i\},\ j = 0, \ldots, T.$$

2. Агент заинтересован в максимизации суммарного выигрыша

$$\varphi(u_j) = \sum_{j}^{T} u_j,\ j = 0, \ldots, T.$$

3. Агент заинтересован в максимизации среднего по времени выигрыша

$$\varphi(u_j) = \frac{1}{T} \sum_{j}^{T} u_j,\ j = 0, \ldots, T.$$

Под коллективом понимают распределенную децентрализованную систему, которая способна изменять свою структуру – самоорганизовываться.

Распределенность здесь – это ограниченность информационного взаимодействия агентов. Например, в пределах заданного ограниченного радиуса видимости средств связи и опознавания соседних агентов.

Децентрализованность – это когда каждый агент самостоятельно принимает и реализует решения в условиях полного или частичного отсутствия единого центра управления.

В полностью децентрализованных системах управление происходит только за счет локальных взаимодействий между агентами. При этом наряду с распределенными знаниями и ресурсами, описываются локальные задачи отдельных агентов, решаемые на базе локальных концептуальных моделей и локальных критериев.

Самоорганизация – это когда система способна изменять свою структуру в зависимости от поставленных перед ней задачами, изменений в окружающей среде и накопленных системой сведений об этих изменениях. При этом общая цель самоорганизации заключается в получении новых функциональных возможностей более высокого порядка, чем функциональные возможности отдельного агента.

Основными характеристиками коллектива агентов является количество агентов и их состав. Количество агентов из которых состоит коллектив – N. Различают маленькие – $N < 10$, средние – $10 < N < 100$, большие – $100 < N < 1000$ и сверх большие – $N > 1000$ коллективы.

Состав коллектива может быть *однородный* или *неоднородный*. Например, у различных агентов могут быть разные наборы доступных действий влияния на среду. Предельный случай: полностью однородный коллектив, когда все характеристики функциональной схемы агентов одинаковы.

3. Связность коллектива это степень интегрированности отдельных агентов в единую систему – коллектив. В дальнейшем будем рассматривать информационную связность, считая, что в других аспектах (конструктивном, функциональном, энергетическом) агенты полностью автономны. Возможны две основные

формы: не самовыявленый коллектив (связь между агентами отсутствует) и самовыявленый коллектив (есть связь между агентами).

Целевая функция коллектива отражает предыдущую историю развития действий в коллективное поведение

$$AG : R \xrightarrow{W(U)} AD,$$

где $AG = \{ag_i\}, i = 1, \ldots, N$ – коллектив агентов; R – множество всех возможных конечных последовательностей взаимодействия агента со средой; AD – коллективное действие (совокупность действий всех агентов коллектива).

Задача коллектива максимизировать значение функции $W(U)$, которая зависит от действий всех агентов. Примеры целевых функций коллектива:

– уничтожение максимального количества целей в заданном районе;

– максимально быстрое прохождение лабиринта;

– прохождение через минное поле с наименьшими потерями;

– формирование правильных геометрических фигур в пространстве;

– перевозка грузов, для которых требуются усилия нескольких роботов грузчиков;

– сбор наиболее полной информации об объекте исследования с помощью мобильных измерительных агентов,

– распределение вычислительных ресурсов программными агентами и т.д.

Способ описания коллективного поведения определяет, какой вид представления выбран для функции $W(u)$, и какой подход выбран для ее анализа и синтеза. Наиболее распространены описания коллективного поведения с помощью: функционального анализа, марковских случайных процессов, теории автоматов, теории игр, теории статистических решений и логики предикатов.

В рамках выбранного способа описания разрабатываются модели коллективного поведения, критерии эффективности коллективного поведения и тестовые задачи. Разработчик коллектива агентов получает от заказчика информацию о целевой функции коллектива и ищет способ отображения целевой функ-

ции коллектива в целевые функции отдельных агентов

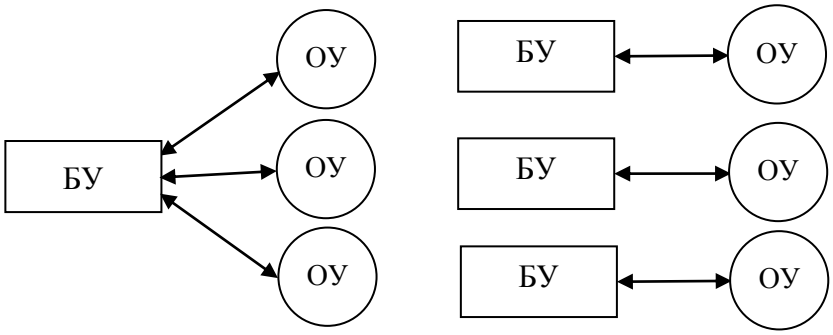
$$W(u) \rightarrow \{\varphi_i(u)\};\ i = 1, \ldots, N,$$

где $\varphi_i(u)$ могут быть разными (неоднородный коллектив). Более сложная задача отобразить $W(u)$ в виде набора одинаковых целевых функций агентов:

$$W(u) \rightarrow \{\varphi(u)\}.$$

Если вид $W(u)$ известен, то решается задача самообучения (изменение параметров функции). В противном случае $W(u)$ функционал и решается задача самоорганизации, поиска вида функции $W(u)$.

Централизованное управление (рис. 9.3, а) предусматривает наличие одного блока управления, который собирает полную информацию обо всех объектах управления и генерирует сигналы управления для всех этих объектов. Децентрализованное управление (рис. 9.3, б) предусматривает, что каждый из объектов управления оснащен собственным блоком управления, который генерирует для него сигналы управления, исходя из информации только о нем.



а) б)

Рис. 9.3. Структура а) централизованного и б) децентрализованного управления (БУ – блок управления, ОУ – объект управления)

К системам управления предъявляются требования по адекватности и оперативности управления. Эти требования противоречат друг другу, поэтому

необходимо искать оптимальное соотношение этих требований. Эти противоречия имеют особый смысл для распределенных систем.

Агент ставится в условия выбора между собственными интересами и интересами коллектива. Это противоречие в поведении агента используется для нахождения коллективом оптимального уровня взаимодействия агентов в заранее неизвестных условиях, что соответствует максимуму целевой функции коллектива.

Противоборство – это случай, когда интересы агентов полностью противоположны. Если интересы агентов не противоположны, то среди них можно выделить общие и противоположные интересы (сотрудничество и соперничество). Возможен сценарий – когда в каждый момент времени N агентов одновременно реализуют выбранные ранее действия и получают отклик среды. В момент реализации действия агенту неизвестны действия других агентов. При этом каждый из них заинтересован, в первую очередь, величиной собственного выигрыша. Рассматривается ситуация, когда индивидуальные выигрыши могут быть различными и действия одних агента могут влиять на величину индивидуального выигрыша других агентов.

Каким же образом агентам соотносить свои индивидуальные выигрыши, чтобы реализовать лучшее коллективное поведение?

Увеличение индивидуального выигрыша отдельного агента не всегда приводит к увеличению коллективного выигрыша. Например при использование ограниченного общего ресурса. Возможный выход в эволюции сотрудничества: долговременное сотрудничество может привести к нелинейному росту суммарного выигрыша, например за счет преодоления ограничений на общий ресурс.

Полезность индивидуальных действий – это оценка вклада данного агента в общий результат коллективных действий. Например, функциональная полезность – объем выполненных работ по сравнению с другими агентами, информационная полезность – количество информации о среде, полученное данным агентом по сравнению с другими. Эта оценка может быть использована

как составляющая индивидуальной функции выигрыша агента по принципу: чем больше полезность действий агента для коллектива, тем больше его выигрыш.

Коллективное знание – это функционально полный набор коллективных моделей и правил их использования, применяемых всеми агентами коллектива. Коллективное знание – не сумма индивидуальных, а качественно новое интегральное свойство. Индивидуальная память агента не способна вместить коллективное знание. Она содержит или какую-то его часть, или отражает его в некотором обобщенном виде. Коллективное знание является целостным единым образованием, оно существует как самостоятельно действующая система, компоненты которой могут исключаться, заменяться и добавляться.

На рис. 9.4 дана схема образования коллективного знания, где 1 – выделение из индивидуальных моделей общих фрагментов; 2 – уточнение и удаление несовпадений; 3 – интеграция индивидуальных специализированных моделей; 4 – формирование коллективных моделей; 5 – оценка и проверка эффективности применения коллективных моделей; 6 – генерирование индивидуальных решений по модификации коллективной модели; 7 – стимулирование индивидуального поведения по развитию коллективной модели; 8 – контроль целостности и корректности коллективных моделей.

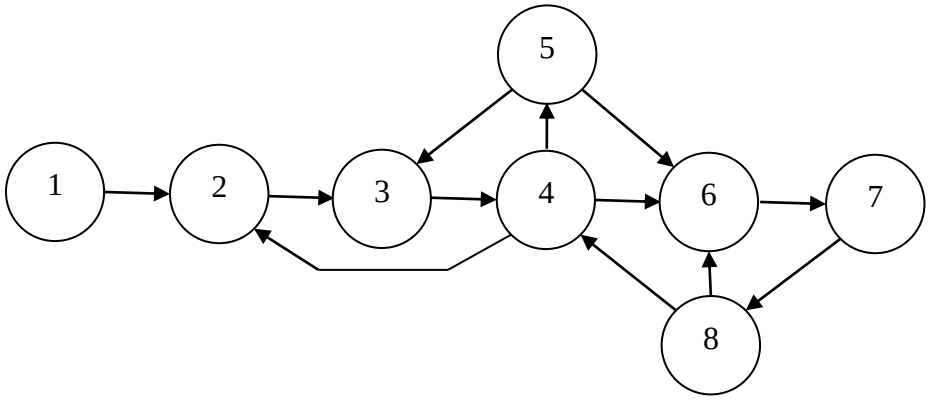


Рис. 9.4. Образование коллективного знания

Коллективный выбор отражает множество индивидуальных предпочтений агентов в одно предпочтение из этого множества, которое в дальнейшем называ-

ется коллективным. Многошаговое повторение коллективного выбора будем называть процессом коллективного принятия решений. В наиболее общем виде проблема коллективного принятия решений заключается в обретении “справедливого” (лучшего по ряду условий) метода объединения индивидуальных предпочтений агентов в единое предпочтение всего коллектива.

Для формальное описание процесса коллективного выбора решений обозначим множество возможных альтернатив как

$$U = \{x_1, x_2, x_3\},$$

а индивидуальный выбор, когда из множества альтернатив выбирается одна, как, например, $\{x_1, x_2, x_3\} \rightarrow x_2$. Тогда индивидуальное предпочтения или порядок предпочтений R^j – это множество альтернатив, которые сортируются в порядке убывания их преимуществ, например: $\{x_1, x_2, x_3\} \rightarrow (x_3, x_1, x_2)$, или $\{x_1, x_2, x_3\} \rightarrow (x_3, x_1 \sim x_2)$, у агента есть две возможности относительно альтернатив – предпочитать или быть равнодушным.

Опишем множество всевозможных индивидуальных предпочтений как

$$R = \{R^1, R^1, \ldots, R^m\},$$

тогда

$$U = \{x_1, x_2, x_3\}, R^1 = (x_1, x_2), R^2 = (x_2, x_1), R^3 = (x_1 \sim x_2), R = \{R^1, R^2, R^3\},$$

это множество всевозможных упорядоченных систем предпочтений

$$R^{(N)} = R \times R \times \ldots \times R.$$

Например

$$N = 2,\ U = \{x_1, x_2, x_3\},\ R = \{R^1, R^2, R^3\},\ R^{(2)} = R \times R.$$

Получаем профиль индивидуальных предпочтений $r_j = (R_1, R_2, \ldots, R_N)$: элемент множества $R^{(N)}$, например

$$N = 2,\ (R^2, R^3).$$

Под функцией коллективного выбора (ФКВ) F будем понимать правило, которое каждому профилю индивидуальных предпочтений ставит в соответствие одно (коллективное) предпочтения из множества R

$$F\colon (R_1, R_2, \ldots, R_N) \to R^j,\ R^j \in R.$$

Область определения F: $R^{(N)}$, область значений F: R.

Функция коллективного выбора может быть справедливой, навязанной, диктаторской, случайной, определенной правилом большинства, и т.п.

Возможен вероятностный подход к принятию коллективного решения (жесткое требование, чтобы ФКВ отображала $R \times R$ в R , а не в распределение вероятностей на R), и возможен случай “сложных вкусов”, когда предпочтения агентов могут иметь несколько градаций силы и слабости.

К решению проблемы коллективного принятия решений применяют следующий подход: вместо того, чтобы принимать решения по отдельным ФКВ, формируется функционально полный набор (система) требований к ФКВ. Каждое из требований отражает свой специфический аспект разумности ФКВ.

Условия, которым должны удовлетворять ФКВ:

1. Универсальность: $N \geq 2$, $m \geq 3$, ФКВ определена для всех возможных $r_j \in R^{(N)}$.

2. Положительная связь коллективных и индивидуальных предпочтений: если ФКВ предпочитает x_i , а не x_j для данного профиля r, то это преимущество должно сохраняться для всех остальных профилей, в которых индивидуальные предпочтения других альтернатив, кроме x_i, не изменяются или изменяются в пользу x_i все соотношения с другими альтернативами.

3. Независимость несвязанных альтернатив: если у нас есть два профиля, такие, что парное сравнение любых двух альтернатив x_i и x_j в этих профилях одинаково для всех агентов, то предпочтения коллектива для двух профилей должны быть также одинаковы.

4. Самостоятельность агентов: для каждой пары альтернатив x_i и x_j, $i \neq j$ существует такой профиль r, что коллектив предпочитает x_i, а не x_j (ФКВ не является навязанною со стороны относительно этой пары альтернатив).

5. Отсутствие диктаторства: не существует такого агента, что если он предпочитает x_i, а не x_j, то коллектив так же предпочитает x_i, а не x_j, независимо от предпочтений других агентов.

Парадокс Эрроу о невозможности состоит в том, что условия 1, 2, 3, 4 и 5 являются несовместимыми, т.к. не существует ФКВ, которая удовлетворяла каждому из этих условий.

Преодолеть парадокс Эрроу возможно с помощью следующих действий:

1. Сохранить формулировку задачи, но отбросить какое-то одно из условий, как слишком жесткое.

2 . Изменить формулировку задачи путем изменения исходных данных или требований ФКВ.

Создано много различных систем, реализующих мультиагентный подход – JADE, Jadex, InterRap и др., которые ориентированы на различные типы организации агентов. Еще одной из таких систем является система AnyLogic, которая предоставляет инструментарий не только для моделирования процесса, но и для визуализации процессов моделирования в 2D и 3D форматах. AnyLogic имеет удобный и понятный интерфейс для создания модели и документацию по работе. Платформа Java в AnyLogic предоставляет безграничную расширяемость моделей за счет программирования на Java. Модели, созданные в этой среде разработки, являются кросс-платформенными [31].

Основной технологией программирования в AnyLogic является визуальное программирование – построение с помощью графических объектов и пиктограмм иерархий структуры и поведения активных объектов [31, 32]. Все объекты, определенные пользователем при разработке модели с помощью графи-

ческого редактора, компилируются в конструкции языка Java, а затем происходит компиляция всей собранной программы на Java, преобразующей модель, в исполняемый код [32].

Основными средствами описания поведения объектов являются переменные, события и диаграммы состояний. Диаграммы состояний позволяют визуально представить поведение объекта во времени под воздействием событий или условий, они состоят из графического изображения состояний и переходов между ними. Любая сложная логика поведения объектов модели может быть выражена с помощью комбинации стейтчартов, дифференциальных и алгебраических уравнений, переменных, таймеров и программного кода на Java [32].

В моделях AnyLogic модельное время может изменяться либо непрерывно, если поведение объектов описывается дифференциальными уравнениями, либо дискретно, переключаясь от момента наступления одного события к моменту наступления следующего события, если в модели присутствуют только дискретные события. Единицу модельного времени разработчик модели может интерпретировать как любой отрезок времени: секунду, час и т.д. [32].

AnyLogic имеет удобные средства представления функционирования моделируемой системы в форме динамической анимации 2D и 3D. Визуализация процесса функционирования моделируемой системы позволяет проверить адекватность модели, выявить ошибки при задании логики. Графические элементы, добавленные на анимацию, называются динамическими, поскольку все их параметры: видимость, цвет и т.п. – можно сделать зависимыми от переменных и параметров модели, которые меняются со временем при выполнении модели. В AnyLogic поддерживается как двумерная, так и трёхмерная анимация.

Крупнейшие компании мира Boeing, Hewlett Packard, IBM, Panasonic, Mitsubishi, NASA, Volvo, McDonald’s управляют своими предприятиями с помощью систем, созданных на основе AnyLogic.

Контрольные вопросы

1. Каковы свойства агентных систем?
2. Что такое интеллектуальный агент?
3. Какова классификация агентов?
4. Каковы базовые этапы моделирования МАС?
5. Какова методика восходящего проектирования МАС?
6. Что характеризуют роли агентов?
7. Какие есть виды агентов?
8. Назовите основные части архитектуры агентной системы.
9. Какие есть основные направлениями в разработке МАС?
10. Перечислите этапы распределенного решения задачи.
11. Какова обобщенная структура агента?
12. Приведите схему централизованного и децентрализованного управления.
13. Что такое целевая функция агента?
14. Что такое целевая функция коллектива?
15. Что такое коллективное знание?

Список рекомендуемой литературы

1. Люгер Д.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем / Д.Ф. Люгер. – М.: Вильямс, 2003. – 864 с.

2. Рутковский Л. Методы и технологии искусственного интеллекта. – М.: Горячая линия, 2010. – 520 с.

3. Bow. S.-T. Pattern Recognition and Image Preprocessing. Dekker, New York-Basel, 2002. – 698 p.

4. Лорьер Ж.-Л. Системы искусственного интеллекта. М.: Мир, 1991. – 568 с.

5. Николенко С.И. Самообучающиеся системы / С.И. Николенко, А. Л. Тулупьев. – М.: МЦНМО, 2009. – 288 с.

6. Марселлус Д. Программирование экспертных систем на Турбо Прологе / Д. Марселлус. – М.: Финансы и статистика, 1994. – 256 с.

7. Уотермен Д. Руководство по экспертным системам. / Д. Уотермен. – М.: Мир, 1980.

8. Джексон П. Введение в экспертные системы: Пер. с англ.: Учебное пособие. / П. Джексон. – М.: Вильямс, 2001. – 624 с.

9. Джарратано Дж. Экспертные системы: принципы разработки и программирование / Дж. Джарратано, Г. Райли. – М.: Вильямс, 2006. – 1152 с.

10. Интернет Университет Информационных Технологий – дистанционное образование [Электронный ресурс] / В.Л. Афонин, Макушкин В.А. Лекция: № 2. Системы представления знаний – 2012 – Режим доступа: http://www.intuit.ru/department/human/isrob/2/3.html

11. Кравець В.О. Вступ до експертних систем / В.О. Кравець, І. П. Хавіна. – Х.: НТУ “ХПІ”, – 2006. – 232 с.

12. Никитина Л.А. Программирование в среде Turbo Prolog: текст лекций / Л. А. Никитина, И. П. Хавина, А. Э. Заволодько. – Х.: НТУ “ХПИ”, 2007. – 100 с.

13. Никитина Л.А. Программирование в среде Turbo Prolog: лабораторный практикум / Л.А. Никитина, И. П. Хавина, А. Э. Заволодько. – Х.: НТУ “ХПИ”, 2007. – 92 с.

14. Розенблатт Ф. Принциипы нейродинамики. – М.: Мир, 1965. – 480 с.

15. Нейрокомпьютеры и интеллектуальные роботы / Анисимов Н.М., Байдык Г.Н., и др. Под ред. Анисимова Н.М. – К.: Наукова думка, 1994. – 272 с.

16. Галушкин А.И. Нейрокомпьтеры и их применение на рубеже тысячелетия в Китае. В 2-х томах. Том 1. – М.: Горячая линия – Телеком, 2004. – 367 с.

17. Хайкин С. Нейронные сети: полный курс / С. Хайкин – М.: Вильямс, 2006. – 1104 с.

18. Дмитриенко В.Д. Основы теории нейронных сетей / В. Д. Дмитриенко, Н. И. Корсунов. – Белгород: БИИММАП, 2001. – 159 с.

19.

20.

21. Фогель Л. Искусственный интеллект и эволюционное моделирование / Л. Фогель, А. Оуэнс, М. Уолш. – М.: Мир, 1969. – 230 с.

22. Ивахненко А.Г. Принятие решений на основе самоорганизации / А.Г. Ивахненко, Ю.П. Зайченко Ю.П. – М.: Советское радио, 1976. – 280 с.

23. Ивахненко А.Г. Помехоустойчивость моделирования / А.Г. Ивахненко, В.С. Степашко – К.: Наукова думка, 1985. – 216 с.

24. Верлань А.Ф. Эволюционные методы компьютерного моделирования / А. Ф. Верлань, В.Д. Дмитриенко, Н.И. Корсунов – К. : Наукова думка, 1992. – 256 с.

25. Гладков Л.А. Генетические алгоритмы / Л.А. Гладков, В. П. Курейчик, В.М. Курейчик. – М.: Физматлит, 2006. – 320 с.

26. Скобцов Ю.А. Основи еволюційних обчислень. Навчальний посібник. Ю.А. Скобцов. – Донецьк: ДонНТУ, 2008. – 326 с.

27. Higuchi T., Iba H., Manderick B. Evolvable Hardware with Genetic Learning // Massively Parallel Artificial Intelligence. – California, Cambridge, London: AAAI Press / The MIT Press, 1994. – P. 398-420.

28. Evolvable Hardware with genetic learning // Higuchi T., Niwa T., Tanaka T., Iba H., de Garis H., Furuya T. // Proceedings of Simulated Adaptive Behavior / The MIT Press, 1993. – P. 67-68.

29. Дюк В. Data Mining: учебный курс / В. Дюк, А. Самойленко. – СПб.: Питер, 2001. – 386 с.

30. Тарасов В.Б. От мультиагентных систем к интеллектуальным организациям: философия, психология, информатика / В.Б. Тарасов. – М.: Едиториал УРСС, 2002. – 352 с.

31. Карпов Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. / Ю. Г. Карпов. – СПб.: БХВ, 2005. – 400 с.

32. Киселёва М.В. Имитационное моделирование систем в среде AnyLogic: учебно-методическое пособие / М.В. Киселёва. – Екатеринбург: УГТУ УПИ, 2009. – 88 с.